LINUX
from Scratch

Gerard Beekmans

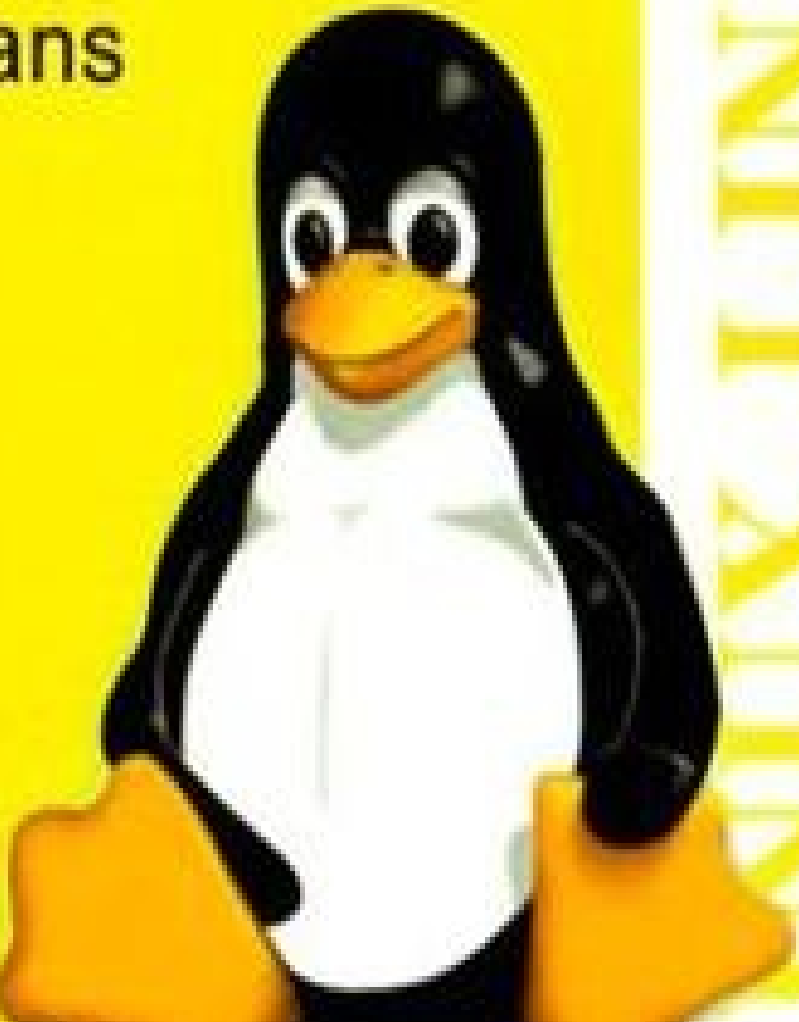


Table of Contents

Linux From Scratch

版本 6.2

Gerard Beekmans

谨以本书献给 LinuxSir.org 以及所有热爱 Linux 的人们。 译者： 金步国(0-5章) ipconfigme(6-7章) bobkey(8-9章) 发布日期：2008年3月1日 [最终正式版]

〔致谢〕感谢之前的 LFS 5.0 和 6.0 翻译小组，没有他们之前辛勤工作积累的资料单靠我们3个人是不可能完成这项工作的。同样也要感谢所有指出预览版中错误的朋友以及对中译本提出建议与期望的朋友(随机顺序)：fisow Robot5 tonytop cnhnln youbest leiv d00m3d asdmusic crandyworld juwen_zhong 晨想 alexlee1216 sonic_yq kikiwarm shooter x_crdjn ilptt linlin911911 ，是你们让中文版更加完美。

〔题外话〕大部分 LFSer 都认为学习 LFS 需要有熟练使用 Linux 的基础，并且大部分听说过 LFS 的人都有一个印象：那是高手的玩具，不是我等菜鸟玩得了的......我不完全赞同，我认为基础如何并非关键，契而不舍的精神和强烈的求知欲才更加重要。想想自己接触 Linux 一个月左右的时候就有了和 Gerard 一样的想法，因为在学习 RedHat / Fedora / Debian 甚至是 Gentoo 的时候，我感觉到自己并不是在学习 Linux 而是在学习这些发行版各自的专有特性，他们把 Linux 本来的面貌层层包裹起来，让我不能深入理解背后的机制。并且这些版本各自有自己的优点和缺点，不能完全满足我的要求。其实那时候我的 Linux 水平仅仅限于会在控制台上敲几个 ls 之类的命令，从未编译过软件，连 make 都没听说过呢。但是我迫切想知道如何定制一个完全适合自己的 Linux 系统，问了好多 Linuxer ，把 Google 搜了个底朝天，也未能得到完整性的答案，唯一让我印象深刻的就是能够容纳在一张软盘上的 babyLinux ，但是它显然太简单，不能满足我的要求。一直郁闷了很久，好不容易机缘巧合，Qoo 兄弟叫我来 LinuxSir.Org 论坛的 LFS 版看看，当时论坛上只有一份不完整的 LFS 6.0 中文版，看完序言后，我激动的跳了起来，欢呼不已！这就是我梦寐以求的东西啊！于是在尚未安装过 LFS 的情况下，静下心来花了十多天时间先完整的翻译了 LFS 6.1 ，又花了5-6天时间，一行命令一行命令地完成了 LFS 的全过程。在学习 LFS 的20天里，我对 Linux 的理解发生了质的飞跃。大约没有人赞同学习 Linux 可以从 LFS 开始，这确实有一定的道理，但是 LFS 教给你的

都是真正的 Linux "基础知识"，并且这些知识可以为将来的进一步学习打下绝对扎实的基础。所以我要用自己的亲身经历鼓励那些刚刚接触 Linux 的新生牛犊勇敢的从 LFS 开始：没有基础不要紧，缺什么补什么！当你把 LFS 做完了，也就脱离"菜鸟"的行列了，用 LFS 给你的强大翅膀，勇敢地继续飞翔吧！

译者十分愿意与他人共享劳动成果，如果你对我的其他翻译作品或者技术文章有兴趣，可以在如下位置查看现有作品的集：

- 金步国作品集 [http://www.jinbuguo.com/]

序言

i. 前言

我在 linux 上的冒险始于1998年，那时我下载并安装了我的第一个发行版。在用它工作了一段时间之后，我发现了很多我认为需要改进的问题。例如，我不喜欢启动脚本的排列顺序、某些程序的默认设置。我尝试过许多不同的发行版来解决这些问题，但是每个发行版都有各自的优点和缺点。最终，我意识到如果我想对我的 Linux 系统完全满意，我必须从头构建我自己的系统。

这是什么意思呢？我决心不用任何预先编译好的软件包，也不用可以安装基本系统的 CD-ROM 或启动盘。我将使用现有的 Linux 系统来开发自己定制的系统。这个"完美的" Linux 系统将拥有各种发行版的优点而没有它们的缺点。开始的时候，这个想法看起来是困难到令人感到畏惧的，但是我仍坚持这个想法，一个符合我特定需求的系统是可以构建起来的，并且不会建立一个标准却不符合我需求的系统。

在处理好诸如循环依赖和编译错误等各种问题之后，我创建了一个定制的 Linux 系统，这个系统功能完整并且适合我个人的需求。这个过程也使得我可以建立精简而紧凑的 Linux 系统，这样的系统比传统的发行版速度更快而且占用的空间更少。我称之为 Linux From Scratch 系统，或简称为 LFS 系统。

当我把我的目标和经验与 Linux 社区的其他成员分享的时候，很显然别人也有同样的想法。这样定制的 LFS 系统不仅可以满足用户的规范和需求，而且也给程序员和系统管理员们提供一了个理想的提高他们 Linux 技能的机会。由于有这样广泛的兴趣和需求，Linux From Scratch 项目诞生了。

这本 *Linux From Scratch* 指导书给读者提供了设计并构建自定义的 Linux 系统的背景知识和过程指导。本书的重点是 Linux From Scratch 这个项目以及使用 LFS 系统带来的好处。用户可以控制系统的所有特征，包括目录布局、脚本设置和安全设置等等。最终的系统将从源代码直接编译生成，用户可以指定在哪里安装、为什么安装以及怎样安装每一个程序。本书使得读者可以完全按照自己的需求定制他们的 Linux 系统，而且使用户对他们的系统有更多的控制权。

希望您在自己的 LFS 系统上工作愉快，享受真正属于你自己的系统所带来的各种好处。

[译者注] "From Scratch"是一个词组，它的意思是"从零做起，白手起家，从无到有"的意思，因此"Linux From Scratch"本质上不应当理解为一个发行版名称。它最贴切的含义应当是一种"方法/思想"：一切从源代码开始的方法/思想。

-- Gerard Beekmans gerard@linuxfromscratch.org

ii. 目标读者

为什么要读这本书呢？有许多原因，最主要的原因是可以学习如何直接从源代码安装一个 linux 系统。许多人也许会问："当你可以下载和安装一个现成的 linux 系统时，为什么要如此麻烦地从源代码开始手动构建一个 linux 系统呢？"这是一个好问题，也是本书存在本节的原因。

LFS 存在的一个重要原因是可以帮助人们学习 linux 系统内部是如何工作的。构建一个 LFS 系统会帮助演示是什么使 linux 运转，各种组件如何在一起互相依赖的工作。最好的事情之一通过这种学习可以获得完全根据自己的需求定制 linux 系统的能力。

LFS 的一个关键的好处是它让用户对于系统有更多的控制，而不是依赖于他人的 linux 实现。在 LFS 的世界里，你自己坐在司机的位置，掌控系统的每一个细节，比如目录布局和启动脚本配置等等。你也能掌控在哪里、为何、以及怎样安装每一个程序。

LFS 的另一个好处是可以创建一个非常小巧的 linux 系统。当安装一个常规的发行版时，人们经常要被迫安装一些可能永远不会用到的程序。这些程序浪费宝贵的磁盘空间，或更糟的是占用 CPU 资源。要构建一个少于100兆(MB)的 LFS 系统并不困难，这比目前大多数的发行版要小很多。这听起来是不是仍然占用太多空间？我们中的一些人专注于创建非常小的嵌入式 LFS 系统。我们成功的构建了一个只运行 Apache 服务器的系统，大约占 8MB 磁盘空间。进一步的缩减能够减至 5MB 或更少。你用一个常规的发行版试试？！这也只是设计你自己的 linux 所带来的好处之一。[译者注]关于如何构建这样的 Apache 服务器系统的详情，请参见 youbest 兄的两篇大作"做一个功能单一，体积小巧的LFS[5M]"和"我们可以做的更小！《功能单一，体积小巧的LFS》续篇[600K]"。此外，本文译者金步国也有一篇文章《DIY一个实用的 mini-LAPP 服务器》[x86版]，详细讲述了如果从源代码编译一个既实用又小巧的 Linux + Apache + PHP + PostgreSQL + OpenSSH + Iptables 服务器，如果你对服务器情有独钟，也很值得参考。

我们可以拿 linux 发行版与快餐店出售的汉堡打比喻，你不能决定自己正在吃的是什么。相反，LFS 没有给您一个汉堡。而是给您一张制作汉堡的配方。用户可以查阅配方，减掉不想要的配料，增加你自己的配料以增强汉堡的口味。当你对配方满意时，开始去制作它，您可以采用任何你喜欢的方式：或烤、或烘、或炸、或焙。

另外一个比方是把 LFS 与建筑房子比较。LFS 提供房子的框架蓝图，但是需要您自己去建筑它。LFS 包含了在这过程中调整计划的自由以及定制满足用户需求的参考。

自己定制 linux 系统的另一个好处是安全性。通过从源码编译整个系统，您能够审查任何东西，打上所有的安全补丁，而不需要等待别人编译好修补安全漏洞的二进制包。除非是您发现并制作的补丁，否则您无法确保新的二进制包一定被正确编译并修正了问题。

Linux From Scratch 的目标是构建一个完整的、可以使用的基础系统。不想构建自己的 linux 系统的读者，不会从本书中获益〔译者不太赞同这句话，借用 d00m3d 的一句忠告："對任何想深入了解的 Linux 愛好者，不論你現用哪個發行版，最少都應該做一次 LFS ，一定會終身受用的。"〕。如果您仅仅想了解计算机启动的时候做了什么，我们推荐您查看 "From Power Up To Bash Prompt" HOWTO 文档，中文版位于 从按下电源开关到bash提示符 ，英文原版位于 *http://axiom.anu.edu.au/~okeefe/p2b/* 或者 linux 文档工程(TLDP)网站 *http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html* 。那个 HOWTO 构建了一个类似本书的系统，但是它的焦点仅仅限制在创建一个能够启动并进入到 BASH 提示符的系统。 想想您的目标，如果您想构建一个实用的 linux 系统并通过这种方式进行学习，那么本书是您的最佳选择。

构建您自己的 LFS 系统的若干原因以上都列出来了。本节只是冰山一角。随着您 LFS 经验的增长，您将会发现 LFS 真正带给您的信息和知识的力量。

iii. 先决条件

创建 LFS 系统并不是一项非常简单的任务。它需要有一定的 Linux 系统管理知识，以便能够解决问题和正确执行命令。作为最低要求，读者必须具备使用命令行(shell)来运行 cp, mv, ls, cd 等命令的能力。我们还希望读者具备使用和安装 Linux 软件的基本知识[非必须]。

因为本书假定读者至少具备了上述技能，各个 LFS 论坛也不太可能涉及上述基础知识，你可能会发现关于上述基础知识的问题在这些论坛中无人理睬，或者仅仅得到一个指向下述 LFS 预备知识的连接。

在构建一个 LFS 系统之前，我们推荐您先阅读下列 HOWTO [看英文吃力的读者不看也没关系]：

- Software-Building-HOWTO *http://www.tldp.org/HOWTO/Software-Building-HOWTO.html*

  这是一份详尽的关于在 Linux 下编译安装"一般的" Unix 软件的指南。

- The Linux Users' Guide *http://www.linuxhq.com/guides/LUG/guide.html*

  这份指南涵盖了各类 Linux 软件的用法。

- The Essential Pre-Reading Hint *http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt*

  这是一份专为 Linux 新手而写的 LFS 提示，包括一份链接列表，这些链接的信息源包含的主题非常广泛而且内容很棒。任何想安装 LFS 的人都应该对这份提示里的许多主题有所了解。

- LFS 初学者注意事项 *http://www.linuxsir.org/bbs/showthread.php?t=231446*

  这是 LinuxSir.Org 上 LFS 版主"终极幻想"专为 LFS 新手而写的注意事项，中文的哦！

iv. 对宿主系统的要求

你的宿主系统应当安装了下列软件，并且不低于指定的版本号。这些要求对于大部分现在的Linux 发行版来说不成问题。另外要注意的是许多发行版会将软件的头文件额外单独打包存放，常见的名称为"<包名称>-devel"或"<包名称>-dev"。如果你的发行版提供了这些包请务必确保已经安装了它们。

- **Bash-2.05a**
- **Binutils-2.12** (不推荐使用大于 2.16.1 的版本，因为尚未经过测试)
- **Bzip2-1.0.2**
- **Coreutils-5.0** (或者 Sh-Utils-2.0, Textutils-2.0, 和 Fileutils-4.1)
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.0**
- **Gcc-2.95.3** (不推荐使用大于 4.0.3 的版本，因为尚未经过测试)
- **Glibc-2.2.5** (不推荐使用大于 2.3.6 的版本，因为尚未经过测试)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Linux Kernel-2.6.x** (必须是 GCC-3.0 以上版本编译的)

  对内核版本的这两个要求是因为：如果宿主系统的内核版本低于 2.6.2 或者不是用 GCC-3.0 以上版本编译的，那么 Binutils 将不能支持线程本地存储(thread-local storage)，同时 NPTL(本地 POSIX 线程库)的测试程序也会出现段错误。

  如果宿主系统的内核版本低于 2.6.2 或者不是用 GCC-3.0 以上版本编译的，您需要安装一个符合上述条件的新内核，然后用该新内核启动宿主系统。有两种方法可以解决这个问题。第一，如果你的 Linux 供应商提供这样的内核，你可以直接安装它；第二，如果你的 Linux 供应商不提供这样的内核或者你不想安装他们提供的内核，你可以自己编译一个内核。关于编译内核和配置引导管理器(假定宿主系统使用 GRUB)的指导，请查看第八章。

- **Make-3.79.1**
- **Patch-2.5.4**
- **Sed-3.0.2**

- **Tar-1.14**

为了确定宿主系统是否满足上述要求，运行下面的命令进行查看：

```
cat > version-check.sh << "EOF"
#!/bin/bash

# Simple script to list version numbers of critical development tools

bash --version | head -n1 | cut -d" " -f2-4
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-4
bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-8
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
gcc --version | head -n1
/lib/libc.so.6 | head -n1 | cut -d" " -f1-7
grep --version | head -n1
gzip --version | head -n1
cat /proc/version | head -n1 | cut -d" " -f1-3,5-7
make --version | head -n1
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1

EOF

bash version-check.sh
```

v. 排版约定

为了易于理解和使用，一些排版约定在本书中通篇使用，本节包含其中一些排版格式的例子。

```
./configure --prefix=/usr
```

这种样式的文本表明应该按照所看到的内容完整无误的输入，除非有另外的说明。这种样式也用在解释部分，来表明引用了哪些命令。

```
install-info: unknown option '--dir-file=/mnt/lfs/usr/info/dir'
```

这种样式(固定宽度的文本)的文本表明它们是屏幕上输出的内容，可能是命令运行的结果，也有可能用来显示文件名，例如： `/etc/ld.so.conf` 。

强调

这种样式的文本在本书中有几种用途，但主要是用来强调重点。

http://www.linuxfromscratch.org/

这种样式用来显示超链接，不管链接的是 LFS 社区内部还是外部的页面，包括 HOWTO 、下载链接和网站。

```
cat > $LFS/etc/group << "EOF"
root:x:0:
bin:x:1:
......
EOF
```

创建配置文件的时候使用这种样式，第一个命令让系统创建 `$LFS/etc/group` 文件，内容是接下来输入的每一行直到文件结束符(EOF)为止。因此，一般就是按照看到的内容整段输入。

`<替代文本>`

这种样式用来显示那些需要输入替代内容的文本或者是复制粘贴的内容。

`[可选文本]`

这种格式用来封装可选的文本。

`passwd(5)`

这种格式用来指向一个特定的手册页(以下简称 "man")。括号中的数字指定 `man` 中特定的章节。例如， `passwd` 有两个 man 页，按照 LFS 安装指令，分别位于 `/usr/share/man/man1/passwd.1` 和 `/usr/share/man/man5/passwd.5` ，这两个手册页的内容并不

相同。本书中使用 `passwd(5)` 指代 `/usr/share/man/man5/passwd.5` 。`man passwd` 将会显示第一个匹配"passwd"的man页，也就是 `/usr/share/man/man1/passwd.1` 。在这个例子中，你可能需要运行 `man 5 passwd` 命令来指定你想要的man页。不过需要提醒一下的是大多数man页并不存在多个页面，也就是说 `man <程序名>` 通常就足够了。

vi. 本书的组织结构

本书分为以下几个部分：

Part I - 简介

Part I 解释了一些安装 LFS 过程中需要注意的重要事项，这部分还提供了本书的一些要点信息。

Part II - 构建前的准备工作

Part II 描述了安装前的准备工作：准备一个新的分区、下载软件包和建立临时编译工具链。

Part III - 构建 LFS 系统

Part III 引导读者逐步建立整个 LFS 系统：一个一个编译安装全部的软件包，配置启动脚本并安装内核。然后，可以在这个 Linux 基本系统上，根据需要编译安装其它软件包来扩展系统。本书的结尾，是一份易用的参考，列出了所有安装了的程序、库和重要的文件，另外还有一份软件包之间的依赖关系列表。

vii. 勘误表

用于构建 LFS 系统的软件经常会被更新和加强。在 LFS 指导书发布以后也经常发布新的安全警告和 bug 修正。要检查本书中的软件包版本或者使用的指令是否需要更新或更改以堵上安全漏洞或修正 bug ，请查看这里：*http://www.linuxfromscratch.org/lfs/errata/6.2/* 在继续前进之前，你应当将这里的更改应用到这本书相应的地方。

I. 简介

1. 简介

1.1. 如何构建一个 LFS 系统?

我们将用一个已安装好的 Linux 发行版(例如 Debian、Mandrake、Red Hat、SuSE)来构建 LFS 系统。这个已存在的 Linux 系统(宿主系统)将作为建立新系统的起点，提供包括编译器、连接器和 Shell 等创建新系统的必要工具。您安装这个发行版的时候，需要选择"development(开发/编程)"选项，以便可以使用这些工具。

另一个选择是使用 Linux From Scratch 的 LiveCD 。这个CD是一个非常好的宿主系统，它包含了构建一个完整 LFS 系统所需要的一切工具，另外还包含了所有的软件包源代码、补丁和一个本书的拷贝。使用这个CD ，你可以不需要任何网络连接或者下载任何额外的东西。要了解更多关于 LFS LiveCD 的信息或者想下载它，请查看：*http://www.linuxfromscratch.org/livecd/* 。

第二章描述了怎样创建一个新的 Linux 本地分区和文件系统，新的 LFS 系统将在该分区上编译和安装。第三章解释了构建一个 LFS 系统需要那些软件包和补丁，以及怎样把它们存放到新文件系统上。第四章讨论了建立一个适当的工作环境。请仔细阅读第四章，因为它讨论了在开始第五章及其后面的步骤之前，开发者需要知道的几个重要问题。

第五章解释了形成一个基本开发套件(或称为工具链)所需的许多软件的安装，这个工具链将被用来构建第六章中的实际系统。其中一些软件包需要解决循环依赖关系：例如，要编译一个编译器，您首先就需要一个编译器。

第五章告诉用户如何第一遍编译工具链，包括 Binutils 和 GCC(第一遍主要的意思就是指这两个核心软件包后面还将第二次安装)。这些软件包里的程序将静态连接以便在使用时独立于宿主系统。接下来的步骤是编译 Glibc ，也就是 C 运行时库。Glibc 将由第一遍建立的工具链程序编译。然后将第二遍编译的工具链动态连接到刚刚编译好的 Glibc 库上。第五章余下的软件包将使用第二遍建立的工具链来编译。完成这些步骤之后，LFS 的安装过程除了正在运行的内核外，将不再依赖于宿主系统。

为了不依赖宿主系统，初看起来我们需要做很多工作。节 5.2,"工具链技术说明"提供了一个完整的技术说明，包括静态连接的程序和动态连接的程序之间的差异。

第六章将构建完整的 LFS 系统。**chroot**(改变root)程序用来进入一个虚拟的环境并开始一个新的 shell，其根目录是 LFS 分区。这非常类似于重启并让内核将 LFS 分区挂载为根分区。系统实际上并没有重启，而是由 **chroot** 代替了，因为建立一个可启动的系统需要做一些现在还不需要做的额外工作。主要的好处在于"虚根环境(chrooting)"允许您在构建 LFS 的同时可以继续使用宿主系统。在等待软件包编译完成的时候，用户可以切换到不同的虚拟控制台(VC)或者 X 桌面，就像平常一样继续使用计算机。

为了完成安装，第七章设置启动脚本，第八章安装内核和启动引导程序。第九章包含在本书之外获得进一步 LFS 体验的信息。在本书中所有步骤都完成之后，计算机就已经准备好重启进入新的 LFS 系统了。

以上就是概略的过程，在接下来的章节和软件包描述中会讨论每一步的细节。看似复杂的项目将详细阐明，随着读者(就是你啦)踏上 LFS 冒险之路，每一件事情都将依序出现。

1.2. 如何提高 LFS 制作的成功率?

新手第一次做 LFS 的时候通常会遇见各种各样的问题，有许多人就此止步了，这不能不说是一个遗憾。热心的 youbest 兄有一篇大作专门针对新手经常遇见的问题，给出了如何提高 LFS 的成功率以及部分问题的解决方法，建议大家在制作过程中参考。

1.3. 在制作中途关闭计算机以后怎么办?

LFS 的制作过程是相当漫长的，特别是对机器不太好的朋友，有时候不得不中途关机，但对一些不太清楚 LFS 的工作原理的朋友，可能重新开机以后无法正确的恢复到先前的工作状态，因此为了能成功的完成 LFS 全过程，不得不持续开机几十个小时。幸好 youbest 兄有一篇大作解决了这个头疼的问题，给出了制作 LFS 过程中各个阶段恢复工作状态的详细方法(适合LFS6.3)，虽然是基于 LFS 6.1 版本的，但仍然具有很实用的参考价值。

1.2. 与上一版本有何不同?

下面是本书自上一个版本到当前版本的更新列表。

升级到:

- Automake 1.9.6
- Bash 3.1
- Binutils 2.16.1
- Bison 2.2
- Coreutils 5.96
- E2fsprogs 1.39
- File 4.17
- Findutils 4.2.27
- Flex 2.5.33
- Gawk 3.1.5
- GCC 4.0.3
- Gettext 0.14.5
- Glibc 2.3.6
- GRUB 0.97
- IANA-Etc 2.10
- IPRoute2 2.6.16-060323
- Less 394
- LFS-Bootscripts 6.2
- Libtool 1.5.22
- Linux 2.6.16.27
- Linux-Libc-Headers 2.6.12.0
- M4 1.4.4

- Man-pages 2.34
- Ncurses 5.5
- Perl 5.8.8
- Procps 3.2.6
- Psmisc 22.2
- Readline 5.1
- Sed 4.1.5
- Shadow 4.0.15
- TCL 8.4.13
- Udev 096
- Vim 7.0
- Zlib 1.2.3

降级到：

- Groff 1.18.1.1

新增加：

- bash-3.1-fixes-8.patch
- Berkeley DB-4.4.20
- bzip2-1.0.3-bzgrep_security-1.patch
- bzip2-1.0.3-install_docs-1.patch
- db-4.4.20-trap-1.patch
- gawk-3.1.5-segfault_fix-1.patch
- gcc-4.0.3-specs-1.patch
- glibc-2.3.6-inotify-1.patch
- glibc-2.3.6-linux_types-1.patch
- groff-1.18.1.1-debian_fixes-1.patch
- inetutils-1.4.2-gcc4_fixes-3.patch
- kbd-1.12-gcc4_fixes-1.patch

- linux-libc-headers-2.6.12.0-inotify-3.patch
- MAN-DB-2.4.3
- mktemp-1.5-add_tempfile-3.patch
- module-init-tools-3.2.2-modprobe-1.patch
- perl-5.8.8-libc-2.patch
- readline-5.1-fixes-3.patch
- tar-1.15.1-gcc4_fix_tests-1.patch
- texinfo-4.8-tempfile_fix-2.patch
- udev-config-6.2
- vim-7.0-fixes-7.patch
- vim-7.0-mandir-1.patch
- vim-7.0-spellfile-1.patch

删除了：

- flex-2.5.31-debian_fixes-3.patch
- gcc-3.4.3-linkonce-1.patch
- gcc-3.4.3-no_fixincludes-1.patch
- gcc-3.4.3-specs-2.patch
- glibc-2.3.4-fix_test-1.patch
- hotplug-2004-09-23
- inetutils-1.4.2-kernel_headers-1.patch
- iproute2-2.6.11-050330-remove_db-1.patch
- Man-1.6b
- mktemp-1.5-add_tempfile-2.patch
- perl-5.8.6-libc-1.patch
- udev-config-4.rules
- vim-6.3-security_fix-1.patch
- zlib-1.2.2-security_fix-1.patch

1.3. 更新日志

这是 6.2 版本的 Linux From Scratch 指导书，2006年8月1日发布。如果本书的版本过期了六个月以上，可能已经有更新更好的版本可供下载了。要获得更新的版本，请访问 *http://www.linuxfromscratch.org/mirrors.html*。

下面是本书自上一个版本到当前版本的详细更新日志。

Changelog Entries:

- August 2, 2006
    - [dnicholson] - Added to the list of acceptable features in ext3 file systems. Thanks to George Gowers.
- August 1, 2006
    - [dnicholson] - Added text describing a potential failure in the E2fsprogs testsuite when there is not enough memory available and suggest enabling swap space to address this. Also added an explicit `swapon` to the Chapter 2 mounting instructions to ensure that the user has enabled their swap space if desired. Thanks to Nathan Coulson and Alexander Patrakov.
    - [dnicholson] - Added text warning that the Udev testsuite will produce messages in the host's logs. Fixes #1846. Thanks to Archaic.
    - [dnicholson] - Finished adding system inotify support. Split the patch so that the syscall functions are part of the Glibc installation. Thanks to Alexander Patrakov for supplying the proper syscall bits.
- July 31, 2006
    - [bdubbs] - Added a patch vim to fix a spellfile download problem. Thanks to Alexander Patrakov.
- July 30, 2006
    - [bdubbs] - Added notes that udev does not recognize a backslash for line continuation.
    - [bdubbs] - Expanded the note in vim to better explain spell files.
- July 29, 2006
    - [bdubbs] - Added a patch to the linux-libc-headers to add the inotify header.
    - [bdubbs] - Added a patch to Berkeley DB to avoid potential program traps.

- July 21, 2006
    - [bdubbs] - Added the existing bash patch to 第五章 to avoid potential custom scripting problems.
    - [bdubbs] - Added grub-0.97-disk_geometry-1.patch.
    - [bdubbs] - Updated to linux-2.6.16.27. Added a note to use the latest kernel version available in the 2.6.16 series.
    - [bdubbs] - Updated vim patch set to level 7.
    - [bdubbs] - Updated the discussion concerning zimezones.
    - [dnicholson] - Added a reminder to check that the virtual kernel file systems are mounted after the description of the revised chroot command.
    - [dnicholson] - Fixed dead link to Linux Driver Model paper on the Device and Module Handling page. Replaced with sysfs paper by the same author. Thanks to Chris Staub and Bryan Kadzban.
- July 18, 2006
    - [bdubbs] - Several textual corrections. Thanks to Chris Staub.
- July 15, 2006
    - [bdubbs] - Added a patch to module-init-tools to correct a possible problem when aliases are specified with regular expressions.
    - [bdubbs] - Updated the kernel to version 2.6.16.26.
    - [bdubbs] - Added sed to correct path to the find program in updatedb after moving find to /bin.
    - [bdubbs] - Updated text concerning test failures in glibc to describe the most recent results.
- July 13, 2006
    - [bdubbs] - Moved the executables: nice, find, kbd_mode, openvt, and setfont to /bin to support boot scripts. Added --datadir=/lib/kbd to kbd's configure so that keyboard data will always be on the root partition.
    - [bdubbs] - Updated text in section 7.9 (Bash Shell 启动文件) to better explain the Xlib example.
- July 12, 20006
    - [bdubbs] - Updated to man-pages-2.34.

  - [bdubbs] - Updated to e2fsprogs-1.39.
  - [dnicholson] - Changed text explaining the installation of the udev-config rules. Thanks to Matthew Burgess.
  - [dnicholson] - Various fixes and additions for examples of custom rules in Udev courtesy of Alexander Patrakov. Added the "Creating custom symlinks" page which includes examples of creating persistent device symlinks, including CD-ROMs. Added a second set of guidelines for creating persistent symlinks for network cards. Other text touch ups on the configuration pages involving Udev. Closes ticket #1818.
  - [bdubbs] - Updated udev-config and bootscripts download location.
  - [dnicholson] - Added commands to create the vi to vim man page symlink in all available languages. Closes ticket #1811. Thanks to Alexander Patrakov.
  - [dnicholson] - Updated to udev-096 and udev-config-20060712. Removed the bug.c program and the cd symlinks script. The cd symlinks will be covered in 第七章 Closes ticket #1804. Thanks to Alexander Patrakov for making the appropriate changes in the Udev rules.
- July 11, 2006
  - [bdubbs] - Changed url for the SBU pages to a generic location.
  - [bdubbs] - Added clarifying text to section 7.9 concerning charmap specifications. Thanks to Dan Nicholson. Closes ticket #1813.
  - [bdubbs] - Moved text in section 5.7 "调整工具链" referencing TCL out of the caution and into its own note so it does not get included later in gcc-pass2. Closes ticket #1822.
  - [bdubbs] - Updated the kernel to version 2.6.16.24. Closes ticket #1808.
- July 10, 2006
  - [dnicholson] - Specified the full path to `modprobe` in the example modprobe rule. Closes ticket #1812.
  - [dnicholson] - Remove the `locale country` command from the heuristic to determine the locale in Bash Shell 启动文件 since it doesn't produce results in all locales.
- July 7, 2006
  - [matt] - Updated module-init-tools download information as it has a new maintainer.

- June 10, 2006
    - [ken] - Added `gettext.sh` to list of programs installed by gettext, similarly `nologin` for shadow, `grub-set-default` for grub, `enc2xs` and `instmodsh` for perl, `slabtop` for procps, `flock` and `tailf` for util-linux, `bootlogd` for sysvinit, `manpath` for man-db, `filefrag` for e2fsprogs. Thanks to Chris Staub for the patch.
- May 31, 2006
    - [matthew] - Upgrade to Linux-2.6.16.19.
    - [matthew] - Upgrade to Man-pages-2.33.
    - [matthew] - Upgrade to Bison-2.2.
    - [matthew] - Upgrade to Coreutils-5.96.
    - [gerard] - Added `tee` to chapter 6's Glibc `make check` so the output can be seen on screen as well as captured in the log file.
- May 30, 2006
    - [matthew] - Removed an out of date comment regarding having to run `pwconv` to reset passwords after enabling password shadowing. Thanks to Chris Staub for the report.
    - [matthew] - Removed `getunimap`, `setlogons`, and `setvesablank` from the list of programs installed by kbd. Thanks to Chris Staub for the patch.
- May 30, 2006
    - [matthew] - Removed `swapdev` from the list of files installed by util-linux. Thanks to Chris Staub for the patch.
- May 27, 2006
    - [jhuntwork] - Remove the 'refer back's in the gcc-pass2 and chapter06/gcc pages. Better organizes the commands and data so that the flow of the book is not lost.
    - [jhuntwork] - Add a note about installing spell files for Vim in a language other than English.
    - [jhuntwork] - Correct Vim's installation of man pages to work well with Man-DB. Patch from Alexander Patrakov and Ag Hatzim.
- May 26, 2006
    - [jhuntwork] - Some version corrections in the vim page.
- May 25, 2006

    - [jhuntwork] - Updated to Vim-7.0. Fixes #1793.
    - [jhuntwork] - Fixed generation of diff's man page. Thanks Randy McMurchy for the report and Ken Moffat for the fix. Fixes #1800.
- May 22, 2006
    - [jim] - Fixed a constant support question asked in #IRC and the mailing lists about shadow's additional sed command for cracklib. Using a complete sed command instead.
- May 15, 2006
    - [archaic] - Updated to udev-config-20060515. This adds the rule to create /dev/usb nodes as well as making the rules files slightly more modular by reorganizing which rules go to which files. This is a very minor update.
    - [archaic] - Updated to man-pages-2.32.
    - [archaic] - Updated to udev-092.
- May 14, 2006
    - [manuel] - Updated SBU and disk usage values.
    - [manuel] - Created packages.ent. Moved data about packages to packages.ent as entities.
- May 12, 2006
    - [archaic] - Updated to linux-2.6.16.16.
- May 9, 2006
    - [manuel] - Updated packages and patches sizes.
- May 8, 2006
    - [archaic] - Made the directory tree creation more concise and removed the extraneous /opt/* hierarchy (it is not required by FHS). Closes ticket #1656.
- May 7, 2006
    - [archaic] - Updated to linux-2.6.16.14.
    - [ken] - Use ext3 filesystem instead of ext2. Resolves ticket #1792.
- May 6, 2006
    - [jhuntwork] - Added MD5 sums for packages and patches. Resolves ticket #1788.
- May 3, 2006
    - [archaic] - Upgraded to linux-2.6.16.13.

  - [jhuntwork] - Updated stripping notes to reflect current findings. Resolves ticket #1657.
  - [archaic] - Updated the bug.c code to avoid USB-related uevent leakage reports.
- May 2, 2006
  - [jhuntwork] - Fixed sanity checks to work after final GCC and changed their format. Resolves ticket #1768.
  - [archaic] - Removed mention of usbfs from the fstab page since it is already covered in BLFS.
  - [archaic] - Updated to man-pages-2.31.
  - [archaic] - Updated to iana-etc-2.10.
  - [archaic] - Updated to tcl8.4.13.
- May 1, 2006
  - [archaic] - Added two seds to avoid symlink problems with Readline during reinstallation. Thanks to Dan and Manuel for the fix and for testing. Fixes ticket #1770.
  - [archaic] - Fixed issue where module-init-tools would not re-install its binaries.
  - [archaic] - Updated to linux-2.6.16.11.
  - [archaic] - Updated to udev-091. Moved to a tarball-based set of udev rules. Updated the bootscripts to support the new udevsettle program.
- April 27, 2006
  - [manuel] - Added SEO Company Canada to donators acknowledgements.
- April 23, 2006
  - [manuel] - Fixed command to change $LFS/tools ownership. Resolves ticket #1780.
- April 22, 2006
  - [manuel] - Revised again the 对宿主系统的要求 page wording and look. Thanks to Bruce Dubbs for the patch. Resolves ticket #1779.
- April 21, 2006
  - [manuel] - Added commands to determine the version of the required packages installed on the host. Thanks to Bruce Dubbs for the commands list and Randy McMurchy for reviewing the wording.

- [manuel] - Alphabetized patches list. Thanks to Justin R. Knierim for the patch.

- April 20, 2006
    - [jhuntwork] - Updated bash to 3.1.17 via an updated patch. Resolves Ticket 1775.
    - [manuel] - Reworded why a 2.6 kernel compiled with GCC-3 is required on the host system.
    - [manuel] - Revised dependencies info. Thanks to Chris Staub for the patch.
- April 19, 2006
    - [jhuntwork] - Added a more detailed list of minimum software requirements. Thanks to Chris Staub for researching these and Alexander Patrakov for suggesting the enhancement. Resolves Ticket 1598.
- April 18, 2006
    - [jhuntwork] - Moved all dependency information to a new page, Appendix C. Appendix C also contains information concerning the build order. While there might need to be a few tweaks yet, this information is complete enough at this point to close out the long-standing ticket #684. Many thanks to Chris Staub, Dan Nicholson and Manuel Canales Esparcia for helping get this finished.
- April 15, 2006
    - [archaic] - Updated to lfs-bootscripts-20060415.
    - [archaic] - Added patch to glibc to fix build errors in packages that include linux/types.h after sys/kd.h.
- April 14, 2006
    - [ken] - Add security patch for tar to address CVE-2006-0300.
    - [archaic] - Upgraded to man-pages-2.29 and linux-2.6.16.5. No command changes.
    - [manuel] - Changed typography conventions. From now, replaceable text is encapsulated inside < >, optional text inside [ ], and library extensions inside { }. Thanks to Bruce Dubbs for the patch.
- April 13, 2006
    - [archaic] - Removed boot logging rule from /etc/syslog.conf and removed the command to move logger to /bin.
    - [archaic] - Added symlink from vim.1 to vi.1.
    - [archaic] - Added chgpasswd to the list of installed files for Shadow.

    - [archaic] - Merged the udev_update branch to trunk.

- April 12, 2006
    - [jhuntwork] - Rewrote section explaining IP Addresses. Thanks Bryan Kadzban and Bruce Dubbs. Resolves Ticket 1663.
    - [jhuntwork] - Added a pointer to GDBM in Berkeley DB page. Also added explanatory text concerning why LFS chose Debian's convention for storing man pages. Thanks to Tushar Teredesai and Alexander Patrakov. Resolves Ticket 1694.
    - [jhuntwork] - Remove symlink of zsoelim to groff's soelim in chapter 6. Man-DB produces a sufficient zsoelim which overwrites the symlink we used to create.
- April 11, 2006
    - [jhuntwork] - Updated bash-3.1 patch. (Ticket 1758)
- April 8, 2006
    - [jhuntwork] - Added a command to create an empty /etc/mtab file early in chapter 6. This avoids testsuite failures in e2fsprogs and possibly other programs that expect /etc/mtab to be present. Explanation from Dan Nicholson, slightly modified. Also merged the 'Creating Essential Symlinks' section with 'Creating passwd, group and log Files'.
- April 6, 2006
    - [manuel] - Placed home page (when available) and full download links for all packages in chapter03/packages.xml.
    - [jhuntwork] - Merged alphabetical branch to trunk.
- April 2, 2006
    - [archaic] - Moved the chowning of /tools to the end of chapter 5 and rewrote note about backing up or re-using /tools. Moved the mounting of kernel filesystems before the package management page and rewrote the page to mount --bind /dev and mount all other kernel filesystems while outside chroot. Rewrote note about re-entering chroot and remounting kernel filesystems. Removed /dev from the list of dirs created in chroot and added it before chroot.
- March 30, 2006
    - [ken] - Correct my erroneous comment about UTF-8 locales in Man-DB. Thanks to Alexander for explaining it.
    - [ken] - upgraded to Linux-2.6.16.1, Iproute2-2.6.16-060323, and Udev-088.
- March 29, 2006

  - [ken] - Upgrade to shadow-4.0.15 and add convert-mans script to convert its UTF-8 man pages. Thanks to Alexander and Archaic for the script and commands. Fixes tickets #1748 and #1750.
- March 22, 2006
  - [archaic] - Updated to lfs-bootscripts-udev_update-20060321.
- March 21, 2006
  - [archaic] - Updated the bootscripts. Removed references to hotplug and the bootscripts udev patch. Removed reference to udevstart. Added text and commands for generating Udev bug reports.
- March 18, 2006
  - [matthew] - Do not run configure manually for iproute2 as it is run automatically by the Makefile. Thanks to Chris Staub for the patch. Fixes ticket #1734.
  - [matthew] - Make bzdiff use mktemp instead of the deprecated tempfile command. Thanks to Chris Staub for the patch. Fixes ticket #1713.
  - [matthew] - Upgrade to flex-2.5.33.
  - [matthew] - Upgrade to readline-5.1.004.
  - [matthew] - Upgrade to bash-3.1.014.
  - [matthew] - Upgrade to psmisc-22.2.
  - [matthew] - Upgrade to file-4.17.
  - [matthew] - Use updated version of the coreutils suppression patch to prevent coreutils version of the su man page from being installed. Fixes #1690.
- March 11, 2006
  - [matthew] - Upgrade to GCC 4.0.3.
- March 8, 2006
  - [matthew] - Upgrade to Man-pages 2.25.
  - [matthew] - Remove an example of poor Udev support as it does not apply to the kernel used in the book. Thanks to Alexander Patrakov.
  - [matthew] - Upgrade to Linux 2.6.15.6.
  - [matthew] - Upgrade to udev-087.
  - [matthew] - Udev's run_program rules require a null device to be present at an early stage, so create one in /lib/udev/devices.

- March 7, 2006
    - [matthew] - Update Udev rules file to load SCSI modules and upload firmware to devices that need it. Improve explanations of device and module handling. Thanks to Alexander Patrakov.
    - [archaic] - Replaced the debian-specific groff patch with an LFS-style patch.
- March 3, 2006
    - [gerard] - Remove -D_GNU_SOURCE from chapter 5 - Patch. Thanks to Dan Nicholson for the patch.
- March 1, 2006
    - [archaic] - Create the Udev directories before creating the symlinks.
    - [jhuntwork] - Added a description of perl configure flags that help perl deal with a lack of groff. Thanks Dan Nicholson.
- February 27, 2006
    - [archaic] - New bash fixes patch adds patch 011 from Bash upstream. Bash patch 010 broke quoting in certain situations.
- February 20, 2006
    - [matthew] - Use non-deprecated format for accessing MODALIAS keys in the Udev rules file, and prevent the "$" from being expanded by the shell.
    - [matthew] - Add patches 009 and 010 from Bash upstream.
    - [matthew] - Upgrade to Man-pages 2.24.
- February 19, 2006
    - [matthew] - Upgrade Perl libc patch to prevent Perl from trying to find headers on the host system. Fixes bug 1695.
    - [matthew] - Expand the Udev module handling rule to run for every subsystem, not just USB.
    - [matthew] - Upgrade to Linux 2.6.15.4.
    - [matthew] - Upgrade to Udev 085.
    - [matthew] - Install Sed's HTML documentation by using --enable-html instead of editing the Makefile. Thanks to Greg Schafer for the report and the fix.
    - [matthew] - Add upstream fixes 001-002 for Readline.

  - [matthew] - Add upstream fixes 001-008 for Bash.
  - [matthew] - Upgrade to Sed 4.1.5.
  - [matthew] - Upgrade to Man-pages 2.23.
  - [matthew] - Upgrade to Coreutils-5.94.
  - [matthew] - Upgrade to DB-4.4.20.
  - [matthew] - Upgrade to Perl-5.8.8, removing the now unneeded vulnerability and DB module patches.
  - [matthew] - Add the verbose parameter to a couple of commands in Linux-Libc-Headers and DB.
  - [matthew] - Create udev specific directories in udev's instructions instead of the more generic creatingdirs.xml. Add "pts" and "shm" directories to `/lib/udev/devices` so that they can be mounted successfully at boot time.
- February 10, 2006
  - [manuel] - Finished the XML indentation plus few tags changes.
- February 8, 2006
  - [matthew] - Rewrite the majority of chapter07/udev.xml to reflect the new configuration for handling dynamic device naming and module loading.
- February 3, 2006
  - [matthew] - Create the `/lib/firmware` directory that can be used by Udev's `firmware_helper` utility.
  - [matthew] - Add descriptions of Udev's helper binaries.
  - [manuel] - Add udev bootscript patch to whatsnew. Removed hotplug from list of packages to download.
  - [ken] - Add udev bootscript patch to list of patches to download.
  - [ken] - Correct the size of the udev tarball.
- February 2, 2006
  - [matthew] - Upgrade to Udev-084 and build all its extras to enable custom rules to be written more easily. Also, change the rules file to handle kernel module loading and patch the udev bootscript to work with this version of udev.
  - [matthew] - Remove the hotplug package and related bootscript. Udev will now handle device creation and module loading.

- [matthew] - Upgrade to Linux-2.6.15.2.

- January 30, 2006
  - [jhuntwork] - Adjust binutils-pass1 so we don't need to hang on to its source directories. Also use 'gcc -dumpmachine' instead of the MACHTYPE var.
  - [jhuntwork] - Various textual corrections. Thanks Chris Staub.
  - [jhuntwork] - Remove unnecessary LDFLAGS variables in binutils pass 1 and 2. Thanks Dan Nicholson.

- January 29, 2006
  - [jhuntwork] - Restore the use of *startfile_prefix_spec.
  - [jhuntwork] - Remove a spurious -i from the perl command when 再次调整工具链. Thanks Dan Nicholson.

- January 26, 2006
  - [jhuntwork] - Modify chapter 6 Glibc's make install command to allow test-installation.pl to run.
  - [jhuntwork] - Adjust chapter 5 binutils to build a static ld-new for use in the chapter 6 readjusting section. Also add some extended sanity checks. These fixes are adapted from DIY-Linux and Greg Schafer. Thanks to Dan Nicholson for the report, as well.
  - [jhuntwork] - Added 'nodump' to commands in the 包管理 section.

- January 25, 2006
  - [jhuntwork] - Remove ppc specific instructions from chapter 6 patch. Cross-LFS can handle non-x86 arch specifics at this point.
  - [jhuntwork] - Fix chapter 6 Glibc's test-installation.pl to test the correct Glibc. Fixes bug 1675. Thanks to Dan Nicholson for the report and Greg Schafer for the fix.
  - [jhuntwork] - Fixed the re-adjusting of the toolchain in chapter 6 so that chapter 6 GCC and Binutils links against the proper Glibc and so that we don't have to keep the binutils directories from chapter 5. Also moved a note about saving the /tools directory to the beginning of chapter 6. Fixes bug 1677. Thanks to Chris Staub, Alexander Patrakov, Greg Schafer and Tushar Teredesai for reporting and resolving this issue.

  - [matthew] - Upgrade coreutils i18n patch to version 2 to fix `sort -n` and add the en_US.UTF-8 locale to improve coreutils' test coverage. Fixes bugs 1688 and 1689. Thanks to Alexander Patrakov.
  - [matthew] - Add information about package management. Thanks to the BLFS project for the text.
- January 24, 2006
  - [matthew] - Upgrade to Groff-1.18.1.1-11.
- January 23, 2006
  - [matthew] - Upgrade to Man-pages 2.21.
  - [matthew] - Upgrade to Psmisc 22.1.
  - [matthew] - Upgrade to Shadow 4.0.14.
  - [matthew] - Install documentation for the Linux kernel. Thanks to Tushar for the report. Fixes bug 1683.
  - [matthew] - Added a patch to enable Perl's DB_File module to compile with the latest version of Berkeley DB. Thanks to Alexander Patrakov for the patch.
- January 20, 2006
  - [jhuntwork] - Added a patch to fix the sprintf security vulnerability in Perl. Thanks to Tim van der Molen for pointing it out.
- January 17, 2006
  - [jhuntwork] - Fixed locale generation for French UTF-8. Thanks to Dan McGhee for the report and Alexander Patrakov for the fix.
- January 10, 2006
  - [ken]: Define YYENABLE_NLS in bison, to resolve a code difference shown up by Iterative Comparison Analysis. Thanks to Greg Schafer.
  - [ken]: Revert my move of mktemp and add a sed to correct gccbug.
- January 7, 2006
  - [ken]: Alter the Perl instructions to always create an /etc/hosts file. This fixes a potential difference in the 'hostcat' recorded in Config_heavy.pl. Thanks to Bryan Kadzban for explaining this.
  - [ken]: Move grep ahead of libtool, so that the latter will correctly reference /bin/grep in references to EGREP.

  - [ken]: Move mktemp ahead of gcc so that gccbug will use mktemp.
  - [ken]: Give Berkeley DB its full name, and remove the '-lpthread' overrides. Also add pointer to BLFS, thanks to Randy McMurchy.
- January 5, 2006
  - [jhuntwork]: Remove mention of news server until we actually have one. Thanks Randy.
  - [jhuntwork]: Initial addition of UTF-8 support. Thanks to Alexander Patrakov.
- January 3, 2006
  - [matt]: Clarify the description of mktemp's --with-libc configure parameter (fixes bug 1667).
  - [matt]: Upgrade to libtool 1.5.22.
  - [matt]: Upgrade to man-pages 2.18.
  - [matt]: Remove the -v flag from the example mkswap command in chapter 2 as it does not affect verbosity (fixes bug 1674).
- December 31, 2005
  - [ken]: Alter installation of Linux Libc asm Headers in chroot, to be repeatable.
- December 23, 2005
  - [jim]: Corrected version on Vim symlink
- December 21, 2005
  - [matt]: Correctly symlink Vim's documentation to /usr/share/doc. Thanks to Jeremy for the report and the fix.
- December 17, 2005
  - [matt]: Pass a valid path to module-init-tools' --prefix configure switch and remove the now unnecessary --mandir switch
  - [matt]: Symlink Vim's documentation to /usr/share/doc. Fixes bug 1610. Thanks to Randy McMurchy for the original report and to Ken and Jeremy for their investigations into the fix.
  - [matt]: Upgrade to psmisc-21.9
  - [matt]: Upgrade to man-pages-2.17
- December 16, 2005

    - [jhuntwork]: Move Procps to before Perl in chapter 6. Perl's testsuite uses 'ps'.
- December 13, 2005
    - [jhuntwork]: Install Tcl's internal headers to /tools/include, allowing us to drop its source directory right away. Origin is Greg Schafer, and thanks to Dan Nicholson for the report (fixes bug 1670).
- December 12, 2005
    - [jhuntwork]: Updated texinfo patch. Fixes segfault issues with texindex. Thanks to Randy McMurchy for the report and Bruce Dubbs and Joe Ciccone for the fix.
- December 11, 2005
    - [jhuntwork]: Upgrade to tcl-8.4.12
    - [jhuntwork]: Upgrade to less-394.
    - [jhuntwork]: Upgrade to readline-5.1. Also removed bash-3.0 and readline-5.0 specific patches.
    - [jhuntwork]: Upgrade to bash-3.1. Also fixed Tcl to work with the new bash version. Thanks to Alexander Patrakov and ultimately, Greg Schafer for the fix.
    - [jhuntwork]: Changed variable used in readline for linking in ncurses. Thanks to Alexander Patrakov for the fix.
- December 9, 2005
    - [matt]: Upgrade to man-pages-2.16
    - [matt]: Upgrade to module-init-tools-3.2.2
    - [matt]: Upgrade to findutils-4.2.27
- December 7, 2005
    - [matt]: Mention the testsuites (or lack of them) for all packages (fixes bug 1664). Thanks to Chris Staub for the report and analysis of affected packages.
- November 26, 2005
    - [matt]: Don't install the Linuxthreads man pages, the POSIX threading API is documented in the man3p section provided by the man-pages package (fixes bug 1660).
    - [matt]: Remove the incorrect note about not having to dump/check a journalled filesystem (fixes bug 1662).
    - [matt]: Upgrade to module-init-tools 3.2.1.

  - [matt]: Prevent installing the internationalized man pages for Shadow's `groups` binary (thanks to Randy McMurchy for the report).
  - [matt]: Upgrade to man-pages 2.14.
  - [matt]: Upgrade to findutils-4.2.26
  - [manuel]: Changed --strip-path to --strip-components in the unpack of module-init-tools-testsuite package.
- November 23, 2005
  - [gerard]: Corrected reference to 'man page' to 'HTML documentation' in chapter 6/sec
- November 18, 2005
  - [manuel]: Fixed the unpack of the module-init-tools-testsuite package.
- November 16, 2005
  - [jhuntwork]: Textual correction concerning gettext in chapter 5 and the use of --disable-shared
- November 15, 2005
  - [archaic]: Changed the chapter 6 Perl Dpager configure option to reflect the new location of the less binary.
- November 14, 2005
  - [jhuntwork]: Only install `msgfmt` from gettext in chapter 5. This is all that is necessary and prevents gettext from trying to pull in unnecessary elements from the host. Thanks to Greg Schafer for pointing this out.
- November 12, 2005
  - [matt]: Improve the heuristic for determining a locale that is supported by both Glibc and packages outside LFS (bug 1642). Many thanks to Alexander Patrakov for highlighting the numerous issues and for reviewing the various suggested fixes.
  - [jhuntwork]: Move sed to earlier in the build.
  - [jhuntwork]: Move m4 to earlier in the build. Thanks Chris Staub.
- November 11, 2005
  - [matt]: Omit running Bzip2's testsuite as a separate step, as `make` runs it automatically (bug 1652).
- November 10, 2005
  - [jhuntwork]: Initial re-ordering of packages. Thanks to Chris Staub (bug 684).

- November 7, 2005
  - [matt]: Install the binaries from Less to /usr/bin instead of /bin (fixes bug 1643).
  - [matt]: Remove the --libexecdir option from both passes of GCC in chapter 5 (fixes bug 1646). Also change the --libexecdir option for Findutils to conform with the /usr/lib/packagename convention already prevalent in the book (fixes bug 1644).
- November 6, 2005
  - [matt]: Remove the optimization related warnings from the toolchain packages (bug 1650).
  - [matt]: Install Vim's documentation to /usr/share/doc/vim-7.0 instead of /usr/share/vim/vim64/doc (bug 1610). Thanks to Randy McMurchy for the report, and Jeremy Huntwork for the fix.
  - [matt]: Stop Udev from killing udevd processes on the host system (fixes bug 1651). Thanks to Alexander Patrakov for the report and the fix.
  - [matt]: Upgrade to Coreutils 5.93.
  - [matt]: Upgrade to Psmisc 21.8.
  - [matt]: Upgrade to Glibc 2.3.6.
- November 5, 2005
  - [matt]: Add a note to the toolchain sanity check in chapter 5 to explain that if TCL fails to build, it's an indication of a broken toolchain (bug 1581).
- November 3, 2005
  - [matt]: Upgrade to man-pages 2.13.
  - [matt]: Correct the instructions for running Module-Init-Tools' testsuite (fixes bug 1597). Thanks to Greg Schafer, Tushar Teredesai and to Randy McMurchy for providing the patch.
- October 31, 2005
  - [matt]: Upgrade to shadow-4.0.13.
  - [matt]: Upgrade to vim-6.4.
  - [matt]: Upgrade to procps-3.2.6.
  - [matt]: Build udev_run_devd and udev_run_hotplugd and alter the udev rules file so that udev once again executes programs in the /etc/dev.d and /etc/hotplug.d directories (fixes bug 1635). Also change the udev rules to prevent udev from

handling the "card" and "dm" devices as these are managed entirely by programs outside of LFS.

- October 29, 2005
    - [matt]: Upgrade to udev-071
    - [matt]: Upgrade to man-pages 2.11.
    - [matt]: Upgrade to coreutils-5.92. This involved removing the DEFAULT_POSIX2_VERSION environment variable as it is no longer required. The testsuite also requires the Data::Dumper module from Perl, so it is now built in chapter05/perl.xml.
- October 22, 2005
    - [archaic]: Upgrade to m4-1.4.4.
- October 21, 2005
    - [matt]: Upgrade to file-4.16.
    - [matt]: Upgrade to man-pages 2.10.
    - [matt]: Upgrade to ncurses 5.5.
- October 15, 2005
    - [matt]: Upgrade to man-pages 2.09.
    - [matt]: Use an updated version of the Udev rules file (fixes bug 1639).
    - [matt]: Add a cdrom group as required by the Udev rules. file
- October 9, 2005
    - [matt]: Emphasise the fact that one must delete the source directory after each package has been installed. Fixes bug 1638. Thanks to Chris Staub.
- October 8, 2005
    - [archaic]: Added patch to fix poor tempfile creation in Texinfo-4.8 that can lead to a symlink attack.
    - [matt]: Upgrade to iproute2-051007.
- October 7, 2005
    - [matt]: Upgrade to gcc-4.0.2.
- October 4, 2005
    - [matt]: Prevent GCC from running the `fixincludes` script in chapter5 pass2 and

chapter 6 (fixes bug 1636). Thanks to Tushar and Greg for their contributions on this issue.

- September 29, 2005
  - [matt]: Add more explicit reader prerequisites (fixes bug 1629).
  - [matt]: Add `-v` to commands that accept it (fixes bug 1612).
- September 26, 2005
  - [matt]: Upgrade to man-pages-2.08.
- September 24, 2005
  - [matt]: Upgrade to gawk-3.1.5.
  - [matt]: Upgrade to man-1.6b.
  - [matt]: Upgrade to util-linux-2.12r.
- September 20, 2005
  - [matt]: Upgrade to bison-2.1.
- September 17, 2005
  - [matt]: Upgrade to udev-070 and remove the unnecessary "udevdir=/dev" parameter.
  - [matt]: Added patch for coreutils to improve echo's POSIX and bash compatibility and to recognise "\xhh" syntax as required by the test suite in udev-069 and later.
- September 15, 2005
  - [archaic]: Added patch for util-linux to prevent a umount vulnerability.
- September 8, 2005
  - [jhuntwork]: Upgrade to groff-1.19.2
- September 6, 2005
  - [ken]: Reworded the glibc text to expect test failures.
- September 5, 2005
  - [ken]: Add patch to fix some of the math tests in glibc.
- September 4, 2005
  - [matt]: Add patch to stop `cfdisk` segfaulting when invoked on devices with partitions that don't contain an ext2, ext3, xfs or jfs filesystem (see bug 1604).
  - [matt]: Upgrade to libtool-1.5.20.

  - [matt]: Upgrade to findutils-4.2.25.
- September 2, 2005
  - [matt]: The optimization flag for util-linux comes from `configure` rather than `MCONFIG`, so adjust the `sed` in order for the segfault fix to actually work.
  - [matt]: Avoid the potential race condition when invoking `find` to remove GCC's fixed headers.
- August 30th, 2005
  - [matt]: Work around a segfault in cfdisk by compiling with -O instead of the default -O2 optimization setting (fixes bug 1604).
  - [matt]: Update the inetutils patch to use the upstream fix for GCC-4.x compilation problems (fixes bug 1602).
  - [matt]: Upgrade to shadow-2.0.12
  - [ken]: Remove **sed -i** commands from gcc-pass2.
- August 28th, 2005
  - [jhuntwork]: Adjusted tar commands in Bash and Glibc chapter 6 builds for consistency
- August 23rd, 2005
  - [matt]: `find` may fail due to a race condition when deleting files. Remove the && construct in chapter05/adjusting.xml so that the rest of the commands for removing fixed headers will be executed (fixes bug 1621).
  - [matt]: Install Udev's documentation relating to configuring rules (fixes bug 1622).
  - [matt]: Upgrade to Man-1.6a.
- August 20th, 2005
  - [matt]: Stop moving some of coreutils' binaries to /bin as they aren't required to be there (fixes bug 1620).
- August 19th, 2005
  - [matt]: Upgrade to Udev-068.
  - [matt]: Upgrade to IANA-etc-2.00.
  - [matt]: Upgrade to file-4.15.
- August 18th, 2005

  - [matt]: Simplify the method for finding where GCC's default specs file and private include directory live. Additionally, don't assume the host's sed supports the -i switch.
  - [ken]: Add a patch to sanitise bzgrep's handling of filenames.
- August 16th, 2005
  - [matt]: Install sed's man page to /usr/share/doc/sed-4.1.4 instead of /usr/share/doc (fixes bug 1600).
  - [matt]: Upgraded to linux-2.6.12.5.
- August 15th, 2005
  - [matt]: Alter the GCC -fomit-frame-pointer sed to protect from multiple invocations (Greg Schafer).
- August 14th, 2005
  - [ken]: Upgrade shadow to 4.0.11.1 with --enable-shadowgrp as advised by Greg Schafer.
  - [matt]: Mention the common libmudflap test failures in GCC (fixes bug 1615).
  - [matt]: Added patch to install documentation for bzip2 (fixes bug 1603).
  - [matt]: Upgrade to linux-2.6.12.4.
  - [matt]: Add sed to chapter05/gcc-pass2 and chapter06/gcc to ensure they get built with -fomit-frame-pointer so it matches the bootstrap build in chapter05/gcc-pass1 (fixes bug 1609).
  - [matt]: Upgrade to udev-067 including a fix for the failing test (bug 1611).
- August 12th, 2005
  - [matt]: Explain that libiconv isn't required on an LFS system (fixes bug 1614).
  - [matt]: Fix ownership of libtool's libltdl data files (fixes bug 1601).
  - [matt]: Change findutils and vim's configure switch explanations to the convention used in the rest of the book (Bug 1613).
  - [matt]: Expand explanation of device node creation at the start of chapter 6.
  - [matt]: Fix incorrect version number for expect's installed library (Bug 1608).
- August 7th, 2005
  - [archaic]: Added note in Shadow regarding building Cracklib from BLFS first.

- August 6th, 2005
    - [matt]: Add texi2pdf to list of Texinfo's installed files.
    - [matt]: Updated Vim's security patch to address the latest modeline vulnerability.
- July 30th, 2005
    - [matt]: Added instructions for installing Bash documentation (Randy McMurchy).
    - [matt]: Remove GCC linkonce patch from chapter03/patches.xml as it's no longer used in the book
- July 29th, 2005
    - [manuel]: Removed the text about defining gvimrc.
- July 28th, 2005
    - [matt]: Add GCC-4 related patch for kbd.
    - [matt]: Add GCC-4 related patch for inetutils.
    - [matt]: Remove the note regarding a known test failure in GRUB. The test no longer fails under GCC-4.
    - [matt]: Add GCC-4 related patch to chapter06 tar.
- July 27th, 2005
    - [matt]: Don't define gvim's configuration file as we don't compile gvim in LFS (Bruce Dubbs).
- July 26th, 2005
    - [matt]: Remove "groups" from the list of programs installed by shadow, as we use the version provided by coreutils instead (Randy McMurchy).
    - [matt]: Updated to mktemp-1.5-add_tempfile-3.patch, which adds license and copyright information to the previous version.
- July 23rd, 2005
    - [matt]: Moved FORMER*CONTRIBUTORS information into the book, so as people can actually see it. The space constraint argument in that file was weak - it only added another 10 lines to a 255 page document (PDF). Now at least we _publically* acknowledge the efforts of previous contributors.
    - [matt]: Updated to man-pages-2.07.
    - [matt]: Updated to zlib-1.2.3.

- July 22nd, 2005
    - [manuel]: Added obfuscate.sh and modified the Makefile to obfuscate e-mail addresses in XHTML output.
- July 21st, 2005
    - [matt]: Add GCC-4 related patches to chapter06 glibc.
    - [matt]: Unset the GCC_INCLUDEDIR variable once it's no longer needed.
- July 19th, 2005
    - [matt]: Removed flex++ from the list of installed files, as it is no longer present (Randy McMurchy)
- July 18th, 2005
    - [matt]: Re-added the explanation of the fixincludes process and rewording where necessary (Chris Staub), and reworded description of the specs patch.
    - [matt]: Remove all host headers brought in via the fixincludes process, not just pthread.h and sigaction.h
- July 17th, 2005
    - [matt]: Slightly adjusted the specs file seds, to prevent multiple seds from adversely affecting them.
    - [matt]: Removed the fixincludes sed from gcc-pass1 as we may need to fix up host's headers. Also reinstate the associated removal of pthread.h and sigthread.h.
- July 16th, 2005
    - [jhuntwork]: Added sed to chapter 5 gcc builds to force the fixincludes to use the headers in /tools and not the host.
    - [jhuntwork]: Removed no_fixincludes and linkonce patches for gcc4. Also removed the command to remove the fixed pthread.h.
    - [jhuntwork]: Fixed adjusting toolchain sed for both chapters 5 and 6.
- July 15th, 2005
    - [matt]: Updated to Linux-2.6.12.3.
    - [matt]: Added a patch to enable tar to build with gcc-4.0.1
    - [matt]: GCC-4.x no longer installs its specs file by default. Alter the toolchain adjustment stage to first dump the specs file where GCC will find it, then alter it.

- [matt]: Added patches for chapter 5's Glibc to build with gcc-4.0.1
- [matt]: Updated to gcc-4.0.1.
- [matt]: Updated to udev-063.

- July 13th, 2005
  - [matt]: Updated to automake-1.9.6.
- July 8th, 2005
  - [matt]: Updated to udev-062.
  - [matt]: Updated to linux-libc-headers-2.6.12.0.
  - [matt]: Updated to linux-2.6.12.2.
  - [matt]: Updated to shadow-4.0.10.
  - [matt]: Updated to iana-etc-1.10.
- July 6th, 2005
  - [archaic]: Pulled the inetutils kernel header patch out again as it is not needed.
  - [matt]: Updated to e2fsprogs-1.38.
  - [matt]: Updated to binutils-2.16.1.
- July 5th, 2005
  - [matt]: Updated to tcl-8.4.11.
  - [matt]: Updated to man-1.6.
  - [matt]: Updated to file 4.14.
  - [matt]: Updated to man-pages 2.05.
- June 12th, 2005
  - [matt]: Upgraded to gettext-0.14.5.
  - [matt]: Upgraded to perl-5.8.7.
  - [matt]: Upgraded to tcl-8.4.10.
  - [matt]: Upgraded to man-pages-2.03.
- May 24th, 2005
  - [jim]: Changed gcc-specs patch to -2.

- May 23nd, 2005
    - [jim]: Changed changelog to use version entities.
- May 22nd, 2005
    - [matt]: Updated to Udev-058.
    - [matt]: Updated to Libtool-1.5.18.
    - [matt]: Updated to Gcc-3.4.4.
    - [matt]: Updated to Binutils-2.16.
- May 15th, 2005
    - [matt]: Updated to Grub 0.97.
    - [matt]: Updated to Libtool 1.5.16.
    - [jim]: Updated to udev 057.
- April 14, 2005
    - [jim]: Updated to man-pages 2.02.
- April 13, 2005
    - [jim]: Updated to glibc 2.3.5.
    - [jim]: Updated to gettext 0.14.4.
- April 12, 2005
    - [manuel]: Small redaction changes.
- April 11, 2005
    - [manuel]: Several tags and text corrections.
- April 6, 2005
    - [jim]: Removed IPRoute2 patch for a sed (Ryan Oliver).

Branch frozen for LFS 6.1 as of April 5, 2005. Some packages and patches updates related with security up to July 9, 2005.

1.4. 资源

1.4.1. FAQ

在构建 LFS 系统的过程中，如果您遇到错误、有什么问题、或者认为书中有错别字，请首先查看在*http://www.linuxfromscratch.org/faq/* 上的常见问答(FAQ) 。

1.4.2. 邮件列表

`linuxfromscratch.org` 服务器为 LFS 项目的开发提供了许多邮件列表，包括主要的开发和支持列表，以及其它内容。如果 FAQ 没有解决你的问题，你可以在 *http://www.linuxfromscratch.org/search.html* 搜索邮件列表。

要了解这些邮件列表的不同、如何订阅邮件列表、邮件列表的存档位置以及其它的相关信息，请访问：*http://www.linuxfromscratch.org/mail.html* 。

1.4.3. IRC

LFS 社区的许多成员在我们社区的 Internet 多线交谈(IRC)网络上为大家提供帮助。在您使用这种方式获取帮助之前，请确定您的问题没有在 LFS FAQ 和邮件列表存档里出现过。您可以在 `irc.linuxfromscratch.org` 找到 IRC 网络，支持频道是 #LFS-support 。 咱们中国人也有自己的IRC，具体细节请查看LFS 中文IRC在线聊天室，内含IRC简明使用帮助。

1.4.4. 参考

关于软件包的附加信息，可以从 *http://www.linuxfromscratch.org/~matthew/LFS-references.html* 获得许多有用的提示。

1.4.5. 镜像站点

LFS 项目在全世界范围内有许多镜像，可以让访问 Web 站点和下载所需的软件包更加快捷便利。请访问 *http://www.linuxfromscratch.org/mirrors.html* 以获得当前可用的镜像站点列表。

1.4.6. 联系信息

请直接将您的问题和意见发送到上面所列的邮件列表。

1.5. 帮助

如果您在本书中遇到困难或者问题，请查看 FAQ 页：*http://www.linuxfromscratch.org/faq/#generalfaq* ，您所遇到的问题往往这里已经有解答了。如果 FAQ 没有解决您的问题，请尝试找到问题的根源，*http://www.linuxfromscratch.org/hints/downloads/files/errors.txt* 上的提示将给您提供一些故障诊断的指导。您还可以到咱们中国人的 LFS 大本营：LinuxSir.Org 的 LFS 论坛寻求帮助。

如果 FAQ 没有解决你的问题，你可以在 *http://www.linuxfromscratch.org/search.html* 搜索邮件列表。

我们也有一个出色的 LFS 社区通过 IRC 和邮件列表为您提供帮助(请参见 节 1.4, "资源")。然而，我们每天都会收到一些只要简单地察看 FAQ 或者搜索邮件列表就可以解决的问题。为了让我们能够为真正不寻常的问题提供帮助，请您自己先搜索一下！仅在你的搜索得不到问题的解决办法的时候，再向我们寻求帮助。为了帮助诊断和解决问题，您需要在帮助请求中包含所有相关信息。

1.5.1. 需要提及的内容

除了一份您所遇到的问题的简短描述外，在任何帮助请求中您还需要附加一些必需的信息：

- 您所使用的 LFS Book 的版本(本书的版本是 6.2)
- 创建 LFS 所用的宿主 Linux 发行版的名称及其版本
- 遇到问题的软件包以及所在的章节
- 确切的错误信息或故障现象描述
- 您是否完全是按本书所说的在做〔强烈建议新手完全按照本书操作〕

注意

不按本书说的做并不意味着我们就不会给您提供帮助，毕竟 LFS 是个性化的选择。提供您在建立过程中所做的任何更改，将有助于我们估计和确定导致您所遇到的问题的可能原因。

1.5.2. Configure 脚本的问题

如果执行 `configure` 脚本的时候，某个地方出错了，请检查 `config.log` 文件，这个文件可能包含 `configure` 过程中没有输出到屏幕的错误，如果您需要请求帮助，请把这里面相关信息也包含进去。

1.5.3. 编译问题

屏幕输出和各种生成的文件内容，对确定产生编译问题的原因都是有用的。`configure` 脚本和运行 `make` 时所产生的屏幕输出也是有帮助的。不需要把所有的输出都包含进来，只需要包含足够的相关信息就行了。下面是一个从 `make` 的屏幕输出中包含所需信息的例子：

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

这种情形下，许多人会仅仅只包含最后一段：

```
make [2]: *** [make] Error 1
```

这对于正确的分析问题是不够的，因为它只是提示出错了，而没有说是什么出错了。在上面例子中，全部内容都应该保存下来，因为它包括了所运行的命令和相关的错误信息。

http://catb.org/~esr/faqs/smart-questions.html 上有一篇很好的文章〔《提问的智慧》(中文版)〕，讲述如何在 Internet 上寻求帮助，请阅读并遵照文章中的提示，将有助于您得到您所需要的答案。

2. 准备一个新分区

2.1. 简介

本章，我们将为 LFS 系统准备一个分区。步骤是创建一个新的磁盘分区并在这个分区上创建文件系统，然后挂载它。

〔译者注〕很多 LFS 新手第一次都搞不清路径问题，加上指导书中又刻意省略了解包、切换目录、清理目录的命令，着实迷糊了不少人，如果你愿意，那么可以参考一下 youbest 兄的一篇大作：手把手教你如何建立自己的Linux系统（LFS速成手册），其中的每个命令和步骤都写的一清二楚。不过，这篇文章是基于 LFS 6.3 版本的，但是它是真正的手把手教程，可以无痛的完成整个 LFS 全部过程，强烈推荐！

2.2. 创建一个新分区

像大多数其他操作系统一样，LFS 通常安装在一个新的专用分区上。如果你有充足的磁盘空间，推荐将 LFS 系统构建在一个新的空白磁盘分区上。当然，LFS 系统(甚至是多个 LFS 系统)也可以安装在现存的某个操作系统所在的分区上，它们完全可以和平共处。这个文档：*http://www.linuxfromscratch.org/hints/downloads/files/lfs_next_to_existing_systems.txt* 解释了怎样实现上面的目标。但是本书只讨论如何在一个新的空白分区上构建LFS系统。

建立一个最小的系统需要 1.3GB 左右的分区，这样才能有足够的空间存储并编译所有的源码包。当然，如果您打算把 LFS 作为您的主 Linux 系统，您可能会在上面安装其它软件，那么您就需要更大的空间(2~3GB)。LFS 系统本身并不占用这么多空间，所需的空间大部分用来为软件编译提供足够的临时空间，编译软件包的时候需要使用大量的临时空间，软件包装好之后这些临时空间可以回收。

因为编译过程中内存(RAM)并不总是够用的，所以最好使用一个小的硬盘分区作为交换空间。内核使用交换空间来存放不常用到的数据，以便为正在运行的进程腾出内存空间。LFS 系统使用的交换分区与宿主系统使用的交换分区可以是同一个，因此当宿主系统已经有交换分区的时候就不必为 LFS 系统再创建一个了。

启动一个磁盘分区程序，例如 `cfdisk` 或者 `fdisk` ，用即将在上面创建新分区的硬盘名字作为命令行选项，比如主IDE硬盘名字就是 `/dev/hda` 。创建一个 Linux 本地分区，需要的话，您还要创建一个交换分区。如果您还不知道如何使用这两个工具的话，请参考 `cfdisk(8)` 或者 `fdisk(8)` 手册页。

请记住新分区的名称(比如 `hda5`)，本书称其为"LFS 分区"，交换分区的名称也要记住，这些分区的名称以后将在 `/etc/fstab` 文件中用到。

2.3. 在新分区上创建文件系统

空白分区建立之后，现在可以在上面创建文件系统了。在 Linux 世界使用最广泛的是 `ext2` 文件系统，但是随着新的大容量硬盘的出现，日志文件系统开始逐渐流行。`ext3` 是一种被广泛使用的基于 `ext2` 的日志文件系统，并且与 E2fsprogs 工具兼容。我们将创建一个 `ext3` 文件系统。您还可以在 *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html*找到创建其它文件系统的指导。

要在 LFS 分区上创建 `ext3` 文件系统，请运行下面的命令：

```
mke2fs -jv /dev/<xxx>
```

用您创建的 LFS 分区的名称替换 `<xxx>` (我们上面的例子里是 `hda5`)。

注意

有些宿主系统在文件系统创建工具(E2fsprogs)中使用了自定义的增强特性。这可能会导致你在第九章重启进入新的 LFS 系统时出现问题。因为这些特性并不被 LFS 安装的 E2fsprogs 支持，你将会得到一个类似于"unsupported filesystem features, upgrade your e2fsprogs"的错误。你可以使用下面的命令来检查你的宿主系统是否使用了自定义的增强特性：

```
debugfs -R feature /dev/<xxx>
```

如果输出的特性不同于： `has_joural` , `dir_index` , `filetype` , `large_file` , `resize_inode` , `sparse_super` 或 `needs_recovery` ，那么就说明你的宿主系统使用了自定义的增强特性。在这种情况下，为了避免后面的问题，请重新编译 E2fsprogs 包，然后用这个重新编译过的工具来创建你将要用来安装 LFS 系统的文件系统：

```
cd /tmp
tar -xjvf /path/to/sources/e2fsprogs-1.39.tar.bz2
cd e2fsprogs-1.39
mkdir -v build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.39
```

如果创建了交换分区，那么还需要用下面的命令进行格式化，如果您使用已有的交换分区，那么就不需要格式化了。

```
mkswap /dev/<yyy>
```

用您创建的交换分区的名称替换 `<yyy>` 。

2.4. 挂载新分区

创建文件系统之后，要让分区可以访问，需要把分区挂载到一个选定的挂载点上。考虑在本书的目的，我们假定文件系统挂载到 `/mnt/lfs` ，但是您也可以选择别的目录。

选定一个挂载点，并指定给 `LFS` 环境变量，请运行命令：

```
export LFS=/mnt/lfs
```

下一步，创建这个挂载点，并挂载 LFS 文件系统，请运行命令：

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
```

用您创建的 LFS 分区名称替换 `<xxx>` 。

如果 LFS 装在多个分区上(比如一个分区用于 `/` 目录，另一个分区用于 `/usr` 目录)，用下面的命令挂载它们：

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext3 /dev/<yyy> $LFS/usr
```

用相应的分区名称替换 `<xxx>` 和 `<yyy>` 。

请确认挂载新分区的时候没有使用太多的限制选项(如 `nosuid` , `nodev` , `noatime` 选项)。运行不带参数的 `mount` 命令看看挂载的 LFS 分区设置了什么选项，如果出现了 `nosuid` , `nodev` , `noatime` 选项之一，您就需要重新挂载这个分区。

如果你使用了交换分区，可以使用下述 `swapon` 命令确保它被启用了：

```
/sbin/swapon -v /dev/<zzz>
```

将 `<zzz>` 替换为正确的交换分区名。

现在工作的空间已经建立好了，接下来要下载所需的软件包。

3. 软件包和补丁

3.1. 简介

本章包含了一个构建基本 Linux 系统需要下载的软件包清单，列出的版本号是已知可以正常工作的版本，本书就是建立在这些软件包之上的。我们强烈建议您不要使用新的版本，因为用于前一个版本的编译安装命令可能并不适用于新的版本。最新版本的软件包也许需要一个与旧版本不同的工作环境，如果并没有配置这样的工作环境，那么软件包就可能会出现问题。

下载位置可能并不总是有效的，如果在本书发布之后，某个软件的下载位置有了变动，Google(*http://www.google.com/*)可以搜索到大多数的软件包。如果 Google 也搜索不到，请尝试 *http://www.linuxfromscratch.org/lfs/packages.html* 上的其它下载手段。

下载的软件包和补丁需要存放到一个构建过程中方便访问的地方，还需要一个工作目录来解压和编译源码包。`$LFS/sources` 既可以用来存储软件包和补丁，也可以作为工作目录。使用这个目录的好处是，所有需要的部件都在 LFS 分区上，构建过程中的所有步骤都可以访问到。

要创建这个目录，在开始下载之前用 *root* 用户登录，并运行下面的命令：

```
mkdir -v $LFS/sources
```

把目录设置为可写和 sticky 模式，这里"Sticky"的意思是虽然某个目录对于多个用户有写入的权限，但这个目录中的文件只有其所有者才能删除。请运行下面的命令使目录可写，并设置 sticky 模式：

```
chmod -v a+wt $LFS/sources
```

3.2. 全部软件包

下载或者用别的方式获得下列软件包：

Autoconf (2.59) - 904KB:

主页：*http://www.gnu.org/software/autoconf/*

下载：*http://ftp.gnu.org/gnu/autoconf/autoconf-2.59.tar.bz2*

MD5和：`1ee40f7a676b3cfdc0e3f7cd81551b5f`

Automake (1.9.6) - 748KB:

主页：*http://www.gnu.org/software/automake/*

下载：*http://ftp.gnu.org/gnu/automake/automake-1.9.6.tar.bz2*

MD5和：`c11b8100bb311492d8220378fd8bf9e0`

Bash (3.1) - 2,475KB:

主页：*http://www.gnu.org/software/bash/*

下载：*http://ftp.gnu.org/gnu/bash/bash-3.1.tar.gz*

MD5和：`ef5304c4b22aaa5088972c792ed45d72`

Bash Documentation (3.1) - 2,013 KB:

下载：*http://ftp.gnu.org/gnu/bash/bash-doc-3.1.tar.gz*

MD5和：`a8c517c6a7b21b8b855190399c5935ae`

Berkeley DB (4.4.20) - 7,767 KB:

主页：*http://dev.sleepycat.com/*

下载：*http://downloads.sleepycat.com/db-4.4.20.tar.gz*

MD5和：`d84dff288a19186b136b0daf7067ade3`

Binutils (2.16.1) - 12,256 KB:

主页：*http://sources.redhat.com/binutils/*

下载：*http://ftp.gnu.org/gnu/binutils/binutils-2.16.1.tar.bz2*

MD5和：`6a9d529efb285071dad10e1f3d2b2967`

Bison (2.2) - 1,052KB:

主页：*http://www.gnu.org/software/bison/*

下载：*http://ftp.gnu.org/gnu/bison/bison-2.2.tar.bz2*

MD5和：`e345a5d021db850f06ce49eba78af027`

Bzip2 (1.0.3) - 654KB:

主页：*http://www.bzip.org/*

下载：*http://www.bzip.org/1.0.3/bzip2-1.0.3.tar.gz*

MD5和：`8a716bebecb6e647d2e8a29ea5d8447f`

Coreutils (5.96) - 4,948KB:

主页：*http://www.gnu.org/software/coreutils/*

下载：*http://ftp.gnu.org/gnu/coreutils/coreutils-5.96.tar.bz2*

MD5和：`bf55d069d82128fd754a090ce8b5acff`

DejaGNU (1.4.4) - 1,056KB:

主页：*http://www.gnu.org/software/dejagnu/*

下载：*http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.4.4.tar.gz*

MD5和：`053f18fd5d00873de365413cab17a666`

Diffutils (2.8.1) - 762KB:

主页：*http://www.gnu.org/software/diffutils/*

下载：*http://ftp.gnu.org/gnu/diffutils/diffutils-2.8.1.tar.gz*

MD5和：`71f9c5ae19b60608f6c7f162da86a428`

E2fsprogs (1.39) - 3,616KB:

主页：*http://e2fsprogs.sourceforge.net/*

下载：*http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.39.tar.gz?download*

MD5和：`06f7806782e357797fad1d34b7ced0c6`

Expect (5.43.0) - 514KB:

主页：*http://expect.nist.gov/*

下载：*http://expect.nist.gov/src/expect-5.43.0.tar.gz*

MD5和： `43e1dc0e0bc9492cf2e1a6f59f276bc3`

File (4.17) - 544KB:

下载：*ftp://ftp.gw.com/mirrors/pub/unix/file/file-4.17.tar.gz*

MD5和： `50919c65e0181423d66bb25d7fe7b0fd`

注意

4.17 版本的 File 软件包在所列的位置可能下载不到，主下载站点的管理员有时候会在新版本发布之后，删除旧的版本。替代的下载位置是 *http://www.linuxfromscratch.org/lfs/download.html#ftp* ，这里可以下载到所需的版本。

Findutils (4.2.27) - 1,097 KB:

主页：*http://www.gnu.org/software/findutils/*

下载： *http://ftp.gnu.org/gnu/findutils/findutils-4.2.27.tar.gz*

MD5和： `f1e0ddf09f28f8102ff3b90f3b5bc920`

Flex (2.5.33) - 680KB:

主页：*http://flex.sourceforge.net*

下载：*http://prdownloads.sourceforge.net/flex/flex-2.5.33.tar.bz2?download*

MD5和： `343374a00b38d9e39d1158b71af37150`

Gawk (3.1.5) - 1,716KB:

主页：*http://www.gnu.org/software/gawk/*

下载：*http://ftp.gnu.org/gnu/gawk/gawk-3.1.5.tar.bz2*

MD5和： `5703f72d0eea1d463f735aad8222655f`

GCC (4.0.3) - 32,208KB:

主页：*http://gcc.gnu.org/*

下载：*http://ftp.gnu.org/gnu/gcc/gcc-4.0.3/gcc-4.0.3.tar.bz2*

MD5和： `6ff1af12c53cbb3f79b27f2d6a9a3d50`

Gettext (0.14.5) - 6,940KB:

主页：*http://www.gnu.org/software/gettext/*

下载：*http://ftp.gnu.org/gnu/gettext/gettext-0.14.5.tar.gz*

MD5和：`e2f6581626a22a0de66dce1d81d00de3`

Glibc (2.3.6) - 13,687KB:

主页：*http://www.gnu.org/software/libc/*

下载：*http://ftp.gnu.org/gnu/glibc/glibc-2.3.6.tar.bz2*

MD5和：`bfdce99f82d6dbcb64b7f11c05d6bc96`

Glibc LibIDN add-on (2.3.6) - 99 KB:

下载：*http://ftp.gnu.org/gnu/glibc/glibc-libidn-2.3.6.tar.bz2*

MD5和：`49dbe06ce830fc73874d6b38bdc5b4db`

Grep (2.5.1a) - 516KB:

主页：*http://www.gnu.org/software/grep/*

下载：*http://ftp.gnu.org/gnu/grep/grep-2.5.1a.tar.bz2*

MD5和：`52202fe462770fa6be1bb667bd6cf30c`

Groff (1.18.1.1) - 2,208KB:

主页：*http://www.gnu.org/software/groff/*

下载：*http://ftp.gnu.org/gnu/groff/groff-1.18.1.1.tar.gz*

MD5和：`511dbd64b67548c99805f1521f82cc5e`

GRUB (0.97) - 950KB:

主页：*http://www.gnu.org/software/grub/*

下载：*ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz*

MD5和：`cd3f3eb54446be6003156158d51f4884`

Gzip (1.3.5) - 324KB:

主页：*http://www.gzip.org/*

下载：*ftp://alpha.gnu.org/gnu/gzip/gzip-1.3.5.tar.gz*

MD5和：`3d6c191dfd2bf307014b421c12dc8469`

Iana-Etc (2.10) - 184KB:

主页：*http://www.sethwklein.net/projects/iana-etc/*

下载：*http://www.sethwklein.net/projects/iana-etc/downloads/iana-etc-2.10.tar.bz2*

MD5和： `53dea53262b281322143c744ca60ffbb`

Inetutils (1.4.2) - 1,019 KB:

主页：*http://www.gnu.org/software/inetutils/*

下载：*http://ftp.gnu.org/gnu/inetutils/inetutils-1.4.2.tar.gz*

MD5和： `df0909a586ddac2b7a0d62795eea4206`

IPRoute2 (2.6.16-060323) - 378 KB:

主页：*http://linux-net.osdl.org/index.php/Iproute2*

下载：*http://developer.osdl.org/dev/iproute2/download/iproute2-2.6.16-060323.tar.gz*

MD5和： `f31d4516b35bbfeaa72c762f5959e97c`

Kbd (1.12) - 618KB:

下载： *http://www.kernel.org/pub/linux/utils/kbd/kbd-1.12.tar.bz2*

MD5和： `069d1175b4891343b107a8ac2b4a39f6`

Less (394) - 286KB:

主页：*http://www.greenwoodsoftware.com/less/*

下载：*http://www.greenwoodsoftware.com/less/less-394.tar.gz*

MD5和： `a9f072ccefa0d315b325f3e9cdbd4b97`

LFS-Bootscripts (6.2) - 24 KB:

下载：*http://www.linuxfromscratch.org/lfs/downloads/6.2/lfs-bootscripts-6.2.tar.bz2*

MD5和： `45f9efc6b75c26751ddb74d1ad0276c1`

Libtool (1.5.22) - 2,856KB:

主页：*http://www.gnu.org/software/libtool/*

下载：*http://ftp.gnu.org/gnu/libtool/libtool-1.5.22.tar.gz*

MD5和： `8e0ac9797b62ba4dcc8a2fb7936412b0`

Linux (2.6.16.27) - 39,886 KB:

主页：*http://www.kernel.org/*

下载：*http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.27.tar.bz2*

MD5和： `ebedfe5376efec483ce12c1629c7a5b1`

注意

Linux 内核更新很快，主要是由于经常发现安全漏洞。除非勘误表上允许使用其他版本的内核，否则应当使用最新版的 2.6.16.x 内核。请不要使用 2.6.17 或更新版本的内核，因为它们可能与启动脚本不兼容。

Linux-Libc-Headers (2.6.12.0) - 2,481 KB:

下载：*http://ep09.pld-linux.org/~mmazur/linux-libc-headers/linux-libc-headers-2.6.12.0.tar.bz2*

MD5和： `eae2f562afe224ad50f65a6acfb4252c`

M4 (1.4.4) - 376KB:

主页：*http://www.gnu.org/software/m4/*

下载：*http://ftp.gnu.org/gnu/m4/m4-1.4.4.tar.gz*

MD5和： `8d1d64dbecf1494690a0f3ba8db4482a`

Make (3.80) - 900KB:

主页：*http://www.gnu.org/software/make/*

下载：*http://ftp.gnu.org/gnu/make/make-3.80.tar.bz2*

MD5和： `0bbd1df101bc0294d440471e50feca71`

Man-DB (2.4.3) - 798KB:

主页：*http://www.nongnu.org/man-db/*

下载：*http://savannah.nongnu.org/download/man-db/man-db-2.4.3.tar.gz*

MD5和： `30814a47f209f43b152659ba51fc7937`

Man-pages (2.34) - 1,760KB:

下载：*http://www.kernel.org/pub/linux/docs/manpages/man-pages-2.34.tar.bz2*

MD5和： `fb8d9f55fef19ea5ab899437159c9420`

Mktemp (1.5) - 69KB:

主页：*http://www.mktemp.org/*

下载：*ftp://ftp.mktemp.org/pub/mktemp/mktemp-1.5.tar.gz*

MD5和： `9a35c59502a228c6ce2be025fc6e3ff2`

Module-Init-Tools (3.2.2) - 166 KB:

主页：*http://www.kerneltools.org/*

下载：*http://www.kerneltools.org/pub/downloads/module-init-tools/module-init-tools-3.2.2.tar.bz2*

MD5和：`a1ad0a09d3231673f70d631f3f5040e9`

Ncurses (5.5) - 2,260KB:

主页：*http://dickey.his.com/ncurses/*

下载：*ftp://invisible-island.net/ncurses/ncurses-5.5.tar.gz*

MD5和：`e73c1ac10b4bfc46db43b2ddfd6244ef`

Patch (2.5.4) - 183KB:

主页：*http://www.gnu.org/software/patch/*

下载：*http://ftp.gnu.org/gnu/patch/patch-2.5.4.tar.gz*

MD5和：`ee5ae84d115f051d87fcaaef3b4ae782`

Perl (5.8.8) - 9,887KB:

主页：*http://www.perl.com/*

下载：*http://ftp.funet.fi/pub/CPAN/src/perl-5.8.8.tar.bz2*

MD5和：`a377c0c67ab43fd96eeec29ce19e8382`

Procps (3.2.6) - 273KB:

主页：*http://procps.sourceforge.net/*

下载：*http://procps.sourceforge.net/procps-3.2.6.tar.gz*

MD5和：`7ce39ea27d7b3da0e8ad74dd41d06783`

Psmisc (22.2) - 239KB:

主页：*http://psmisc.sourceforge.net/*

下载：*http://prdownloads.sourceforge.net/psmisc/psmisc-22.2.tar.gz?download*

MD5和：`77737c817a40ef2c160a7194b5b64337`

Readline (5.1) - 1,983KB:

主页：*http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*

下载：*http://ftp.gnu.org/gnu/readline/readline-5.1.tar.gz*

MD5和： `7ee5a692db88b30ca48927a13fd60e46`

Sed (4.1.5) - 781KB:

主页：*http://www.gnu.org/software/sed/*

下载：*http://ftp.gnu.org/gnu/sed/sed-4.1.5.tar.gz*

MD5和： `7a1cbbbb3341287308e140bd4834c3ba`

Shadow (4.0.15) - 1,265KB:

下载： *ftp://ftp.pld.org.pl/software/shadow/shadow-4.0.15.tar.bz2*

MD5和： `a0452fa989f8ba45023cc5a08136568e`

注意

4.0.15 版本的 Shadow 可能在这个位置下载不到。主下载站点的管理员有时候会在新版本发布之后，删除旧的版本。替代的下载位置是 *http://www.linuxfromscratch.org/lfs/download.html#ftp* ，这里可以下载到所需的版本。

Sysklogd (1.4.1) - 80KB:

主页：*http://www.infodrom.org/projects/sysklogd/*

下载：*http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.4.1.tar.gz*

MD5和： `d214aa40beabf7bdb0c9b3c64432c774`

Sysvinit (2.86) - 97KB:

下载：*ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/sysvinit-2.86.tar.gz*

MD5和： `7d5d61c026122ab791ac04c8a84db967`

Tar (1.15.1) - 1,574KB:

主页：*http://www.gnu.org/software/tar/*

下载：*http://ftp.gnu.org/gnu/tar/tar-1.15.1.tar.bz2*

MD5和： `57da3c38f8e06589699548a34d5a5d07`

Tcl (8.4.13) - 3,432KB:

主页：*http://tcl.sourceforge.net/*

下载：*http://prdownloads.sourceforge.net/tcl/tcl8.4.13-src.tar.gz?download*

MD5和： `c6b655ad5db095ee73227113220c0523`

Texinfo (4.8) - 1,487KB:

主页：*http://www.gnu.org/software/texinfo/*

下载：*http://ftp.gnu.org/gnu/texinfo/texinfo-4.8.tar.bz2*

MD5和：`6ba369bbfe4afaa56122e65b3ee3a68c`

Udev (096) - 190KB:

主页：*http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html*

下载：*http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-096.tar.bz2*

MD5和：`f4effef7807ce1dc91ab581686ef197b`

Udev Configuration Tarball - 4 KB:

下载：*http://www.linuxfromscratch.org/lfs/downloads/6.2/udev-config-6.2.tar.bz2*

MD5和：`9ff2667ab0f7bfe8182966ef690078a0`

Util-linux (2.12r) - 1,339 KB:

下载：*http://www.kernel.org/pub/linux/utils/util-linux/util-linux-2.12r.tar.bz2*

MD5和：`af9d9e03038481fbf79ea3ac33f116f9`

Vim (7.0) - 6,152KB:

主页：*http://www.vim.org*

下载：*ftp://ftp.vim.org/pub/vim/unix/vim-7.0.tar.bz2*

MD5和：`4ca69757678272f718b1041c810d82d8`

Vim (7.0) language files (optional) - 1,228 KB:

主页：*http://www.vim.org*

下载：*ftp://ftp.vim.org/pub/vim/extra/vim-7.0-lang.tar.gz*

MD5和：`6d43efaff570b5c86e76b833ea0c6a04`

Zlib (1.2.3) - 485KB:

主页：*http://www.zlib.net/*

下载：*http://www.zlib.net/zlib-1.2.3.tar.gz*

MD5和：`debc62758716a169df9f62e6ab2bc634`

这些软件包总的大小为 180 MB

3.3. 需要的补丁

除了下载软件包之外，还有一些补丁需要下载。这些补丁修正了本该由软件包开发者修正的错误；另外也做了一些小的修改，使得软件包之间可以更好的协同工作。构建 LFS 系统需要下列补丁：

Bash Upstream Fixes Patch - 23 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/bash-3.1-fixes-8.patch*

MD5和：`bc337045fa4c5839babf0306cc9df6d0`

Bzip2 Bzgrep Security Fixes Patch - 1.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/bzip2-1.0.3-bzgrep_security-1.patch*

MD5和：`4eae50e4fd690498f23d3057dfad7066`

Bzip2 Documentation Patch - 1.6 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/bzip2-1.0.3-install_docs-1.patch*

MD5和：`9e5dfbf4814b71ef986b872c9af84488`

Coreutils Internationalization Fixes Patch - 101 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/coreutils-5.96-i18n-1.patch*

MD5和：`3df2e6fdb1b5a5c13afedd3d3e05600f`

Coreutils Suppress Uptime, Kill, Su Patch - 13 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/coreutils-5.96-suppress_uptime_kill_su-1.patch*

MD5和：`227d41a6d0f13c31375153eae91e913d`

Coreutils Uname Patch - 4.6 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/coreutils-5.96-uname-1.patch*

MD5和：`c05b735710fbd62239588c07084852a0`

Database (Berkeley) Upstream Fixes Patch - 3.8 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/db-4.4.20-fixes-1.patch*

MD5和：`32b28d1d1108dfcd837fe10c4eb0fbad`

Diffutils Internationalization Fixes Patch - 18 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/diffutils-2.8.1-i18n-1.patch*

MD5和： `c8d481223db274a33b121fb8c25af9f7`

Expect Spawn Patch - 6.8 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/expect-5.43.0-spawn-1.patch*

MD5和： `ef6d0d0221c571fb420afb7033b3bbba`

Gawk Segfault Patch - 1.3 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/gawk-3.1.5-segfault_fix-1.patch*

MD5和： `7679530d88bf3eb56c42eb6aba342ddb`

GCC Specs Patch - 15 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/gcc-4.0.3-specs-1.patch*

MD5和： `0aa7d4c6be50c3855fe812f6faabc306`

Glibc Linux Types Patch - 1.1 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/glibc-2.3.6-linux_types-1.patch*

MD5和： `30ea59ae747478aa9315455543b5bb43`

Glibc Inotify Syscall Functions Patch - 1.4 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/glibc-2.3.6-inotify-1.patch*

MD5和： `94f6d26ae50a0fe6285530fdbae90bbf`

Grep RedHat Fixes Patch - 55 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/grep-2.5.1a-redhat_fixes-2.patch*

MD5和： `2c67910be2d0a54714f63ce350e6d8a6`

Groff Debian Patch - 360 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/groff-1.18.1.1-debian_fixes-1.patch*

MD5和： `a47c281afdda457ba4033498f973400d`

GRUB Disk Geometry Patch - 28 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/grub-0.97-disk_geometry-1.patch*

MD5和： `bf1594e82940e25d089feca74c6f1879`

Gzip Security Patch - 2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/gzip-1.3.5-security_fixes-1.patch*

MD5和： `f107844f01fc49446654ae4a8f8a0728`

Inetutils GCC-4.x Fix Patch - 1.3 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/inetutils-1.4.2-gcc4_fixes-3.patch*

MD5和： `5204fbc503c9fb6a8e353583818db6b9`

Inetutils No-Server-Man-Pages Patch - 4.1 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/inetutils-1.4.2-no_server_man_pages-1.patch*

MD5和： `eb477f532bc6d26e7025fcfc4452511d`

Kbd Backspace/Delete Fix Patch - 11 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/kbd-1.12-backspace-1.patch*

MD5和： `692c88bb76906d99cc20446fadfb6499`

Kbd GCC-4.x Fix Patch - 1.4 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/kbd-1.12-gcc4_fixes-1.patch*

MD5和： `615bc1e381ab646f04d8045751ed1f69`

Linux kernel UTF-8 Composing Patch - 11 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/linux-2.6.16.27-utf8_input-1.patch*

MD5和： `d67b53e1e99c782bd28d879e11ee16c3`

Linux Libc Headers Inotify Patch - 4.7 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/linux-libc-headers-2.6.12.0-inotify-3.patch*

MD5和： `8fd71a4bd3344380bd16caf2c430fa9b`

Mktemp Tempfile Patch - 3.5 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/mktemp-1.5-add_tempfile-3.patch*

MD5和： `65d73faabe3f637ad79853b460d30a19`

Module-init-tools Patch - 1.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/module-init-tools-3.2.2-modprobe-1.patch*

MD5和： `f1e452fdf3b8d7ef60148125e390c3e8`

Ncurses Fixes Patch - 8.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/ncurses-5.5-fixes-1.patch*

MD5和： `0e033185008f21578c6e4c7249f92cbb`

Perl Libc Patch - 1.1 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/perl-5.8.8-libc-2.patch*

MD5和： `3bf8aef1fb6eb6110405e699e4141f99`

Readline Upstream Fixes Patch - 3.8 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/readline-5.1-fixes-3.patch*

MD5和： `e30963cd5c6f6a11a23344af36cfa38c`

Sysklogd 8-Bit Cleanness Patch - 0.9 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/sysklogd-1.4.1-8bit-1.patch*

MD5和： `cc0d9c3bd67a6b6357e42807cf06073e`

Sysklogd Fixes Patch - 27 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/sysklogd-1.4.1-fixes-1.patch*

MD5和： `508104f058d1aef26b3bc8059821935f`

Tar GCC-4.x Fix Patch - 1.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/tar-1.15.1-gcc4_fix_tests-1.patch*

MD5和： `8e286a1394e6bcf2907f13801770a72a`

Tar Security Fixes Patch - 3.9 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/tar-1.15.1-security_fixes-1.patch*

MD5和： `19876e726d9cec9ce1508e3af74dc22e`

Tar Sparse Fix Patch - 0.9 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/tar-1.15.1-sparse_fix-1.patch*

MD5和： `9e3623f7c88d8766878ecb27c980d86a`

Texinfo Multibyte Fixes Patch - 1.5 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/texinfo-4.8-multibyte-1.patch*

MD5和： `6cb5b760cfdd2dd53a0430eb572a8aaa`

Texinfo Tempfile Fix Patch - 2.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/texinfo-4.8-tempfile_fix-2.patch*

MD5和： `559bda136a2ac7777ecb67511227af85`

Util-linux Cramfs Patch - 2.8 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/util-linux-2.12r-cramfs-1.patch*

MD5和： `1c3f40b30e12738eb7b66a35b7374572`

Vim Upstream Fixes Patch - 42 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/vim-7.0-fixes-7.patch*

MD5和： `d274219566702b0bafcb83ab4685bbde`

Vim Man Directories Patch - 4.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/vim-7.0-mandir-1.patch*

MD5和： `b6426eb4192faba1e867ddd502323f5b`

Vim Spellfile Patch - 1.2 KB:

下载：*http://www.linuxfromscratch.org/patches/lfs/6.2/vim-7.0-spellfile-1.patch*

MD5和： `98e59e34cb6e16a8d4671247cebd64ee`

所有补丁总计约 773.8 KB

除了上面列出的必需补丁外，LFS 社区还创建了许多可选的补丁。这些可选补丁修正了某些不重要的问题，或者是开启了某些默认没有开启的功能。请仔细阅读位于 *http://www.linuxfromscratch.org/patches/* 的补丁数据库，选择任何符合您需要的额外补丁。

4. 最后的准备工作

4.1. 关于环境变量 $LFS

环境变量 `LFS` 的使用贯穿全书。保持 `LFS` 总是已定义是很重要的，它应该被设置为 LFS 分区的挂载点。用下面的命令检查它的设置是否恰当：

```
echo $LFS
```

请确认输出的是 LFS 分区挂载点的路径，如果您遵循我们的例子，则应该是 `/mnt/lfs` 。如果输出结果不正确，用下述命令设置它：

```
export LFS=/mnt/lfs
```

设置这个环境变量的好处是您以后可以按照原样输入形如 **mkdir $LFS/tools** 的命令，当 shell 处理这个命令行的时候，会自动用"/mnt/lfs"(或者该环境变量所设定的值)替换"$LFS" 。

当您离开然后重新进入当前工作环境的时候(像 **su** 到 `root` 或者其他用户)，不要忘记检查 `$LFS` 是否设置正确。

4.2. 创建 $LFS/tools 目录

第五章中编译的所有程序都将安装到 `$LFS/tools` 目录下，以便与第六章中编译的程序隔离开来。这里编译的程序只是临时使用的工具，不是最终 LFS 系统的组成部分。把这些程序放到一个单独的目录下，在使用过之后简单的删除掉就可以了。这样做也可以防止这些程序与宿主系统中相同的程序混淆(第五章中容易出现这样的事情)。

以 *root* 用户运行下面的命令来创建所需的目录：

```
mkdir -v $LFS/tools
```

下一步是在宿主系统上创建一个 `/tools` 符号链接，指向 LFS 分区上新创建的目录，这个命令同样要作为 *root* 用户运行：

```
ln -sv $LFS/tools /
```

注意

上述命令是正确的，`ln` 命令的语法有了一点变化。如果您认为发现了一个错误，在报告它之前，先用 `info coreutils ln` 和 `ln(1)` 命令查看手册页。

所创建的符号链接使得将要编译的工具链总是位于 `/tools` 目录下，这意味着编译器、汇编器和连接器在本章(这时我们还在使用宿主系统的一些工具)和下一章(这时我们"chrooted"到了 LFS 分区)都可以使用。

4.3. 添加 LFS 用户

以 *root* 用户登录的时候，犯一个错误就足以损坏甚至摧毁系统，因此，本章我们推荐使用一个无特权的用户来编译和安装软件包。您可以使用您自己的用户名，不过为了建立一个干净的工作环境，建议您新建一个名为 *lfs* 的组，并在其中添加一个名为 *lfs* 的用户，我们将在安装过程中使用这个用户。以 *root* 用户运行下列命令来添加新用户：

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```

命令行选项的含义：

`-s /bin/bash`

指定 `bash` 作为 *lfs* 用户的默认 shell

`-g lfs`

将 *lfs* 用户添加到 *lfs* 组

`-m`

为 *lfs* 用户创建 home 目录

`-k /dev/null`

这个参数通过修改输入位置为特殊的空设备来防止从框架目录(默认为 `/etc/skel`)拷贝文件

`lfs`

这是所创建的组和用户的实际名字

为了可以使用 `lfs` 用户登录(与从 `root` 用户切换到 `lfs` 用户不同，这种切换不需要 `lfs` 用户有一个密码)，必须先为 `lfs` 用户设置一个密码：

```
passwd lfs
```

通过把 `lfs` 用户设置为 `$LFS/tools` 目录的所有者来授予 *lfs* 用户对该目录的完全访问权限：

```
chown -v lfs $LFS/tools
```

如果您依照建议创建了独立的工作目录，请将该目录的所有权赋予 `lfs` 用户：

```
chown -v lfs $LFS/sources
```

接下来，以 `lfs` 用户登录。您可以通过一个虚拟控制台，或者通过显示管理器，或者用下面的切换用户命令：

```
su - lfs
```

" `-` " 参数指示 `su` 开启一个登陆Shell(login shell)而不是非登陆Shell(non-login shell)，这两种 shell 的不同之处请参考 `bash(1)` 和 `info bash` 。

4.4. 设置工作环境

通过给 `bash` shell 创建两个新的启动文件来设置一个良好的工作环境。用 *lfs* 用户登录，输入下面的命令来创建一个新的 `.bash_profile` 文件：

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

作为 *lfs* 用户登录的时候，初始 shell 通常是一个登陆_*shell(_login* shell)，它会首先读取宿主系统的 `/etc/profile` 文件(可能包含一些设置和环境变量)，然后继续读取 `.bash_profile` 文件来完成登录初始化。`.bash_profile` 文件中的 `exec env -i.../bin/bash` 命令用完全空的环境来取代当前的环境（除了继承 `HOME`, `TERM`, `PS1` 变量外）。这样能保证我们的编译环境不会被宿主系统中不必要的或者有潜在危险的环境变量所影响，从而确保获得一个干净的工作环境。

另一个新的 shell 实例是非登陆_*shell(_non-login* shell)，它不读取 `/etc/profile` 或 `.bash_profile` 文件，而是读取 `.bashrc` 文件。现在创建 `.bashrc` 文件：

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL PATH
EOF
```

`set +h` 命令关闭 `bash` 的 hash 功能，hash 通常是一个有用的特性：`bash` 使用一个 hash 表来记录可执行文件的完整路径，以避免为了找到同一个可执行文件而反复搜索 `PATH` 里的目录。然而，新工具装好之后就要立即使用，通过关闭 hash 功能，当要运行程序的时候，shell 将总是搜索 `PATH` 里的目录，这样新工具一编译好，shell 就可以在 `$LFS/tools` 目录里找到，而不是执行所记忆的其它地方的旧版本程序。

将用户文件创建掩码(umask)设为 022 ，使得新创建的文件和目录只有所有者可写，其他用户只能读取和运行(`open(2)` 系统调用的默认模式是新文件权限 644，新目录权限 755)。

`LFS` 环境变量应该设为所选择的挂载点。

`LC_ALL` 环境变量控制着某些程序的本地化，使其显示的信息遵循指定国家的惯例。如果宿主系统使用的 Glibc 版本低于 2.2.4，将 `LC_ALL` 环境变量设置为"POSIX"或"C"以外的值(在本章中)可能会在您退出虚根环境后再想返回的时候出现问题。请把 `LC_ALL` 设置为"POSIX"或"C"(这两者是等价的)以确保在虚根环境中的所有东西都像预期的那样正常工作。

通过把 `/tools/bin` 放在 `PATH` 的最前面，第五章中所有的程序安装好后，就可以立即被 shell 运行。将这一点和关闭 hash 功能结合起来，预防了宿主系统的旧程序在不该运行的时候却被运行了的风险。

最后，为了完全准备好编译临时工具的工作环境，导入刚刚创建的 profile 文件：

```
source ~/.bash_profile
```

4.5. 关于 SBU

许多人都想知道编译和安装一个软件包预计需要多长时间。因为 Linux From Scratch 可以在多种不同的系统上创建，准确估计所需的时间是不可能的。最快的系统上编译安装最大的软件包(Glibc)大约需要 20 分钟，但在很慢的系统上可能耗费长达三天时间。我们不提供准确时间，代之以标准编译时间单位(SBU)来度量。

SBU 度量具体说明如下，本书中第一个编译的软件包是第五章中静态编译的 Binutils 。编译这个软件包所花费的时间就作为标准编译时间单位(SBU)。所有其它软件的编译时间都用这个时间来衡量。

例如，对于一个编译时间为 4.5 SBU 的软件包，这意味着如果一个系统静态编译安装 Binutils 需要花费 10 分钟，那么编译这个软件包将大约需要 45 分钟。幸运的是，大多数软件包编译安装所需的时间都比 Binutils 所需的时间要短。

SBU并不十分精确，因为它依赖于许多因素，包括宿主系统 GCC 的版本。另外，在基于对称多处理器(SMP)的机器上，SBU 更加不准确。我们提供 SBU，仅仅是给出安装一个软件包所需时间的大概估计，在某些情况下实际花费的时间与预估计的时间之间可能有数十分钟的差异。

要查看在一些特定机器上的实际编译安装时间，我们推荐您查看 LinuxFromScratch SBU 的主页 *http://www.linuxfromscratch.org/~bdubbs/* 。

4.6. 关于软件包测试套件

大多数软件包提供了测试程序集。编译完一个软件包之后立即运行一下它提供的测试程序是个好主意，因为测试程序将进行一次"健全检查"来确认所有的代码是否都正确编译了。通过了测试程序集的一系列检查，通常意味着软件包在按照开发者的预期工作，但是这并不保证这个软件包就完全没有 bug 了。

某些软件包的测试程序是极为重要的，例如核心工具链软件包 GCC 、Binutils 、Glibc 的测试程序，这是因为这些软件包在一个良好工作的系统里的处于核心角色。GCC 和 Glibc 的测试程序需要运行很长时间，尤其在速度慢的硬件上，但我们仍然强烈建议您运行这些测试程序。

注意

经验显示，运行第五章中的测试程序时会有点小问题。不可避免的是，宿主系统总是对这一章的测试程序施加某些影响，常常导致无法解释的测试失败。因为第五章中编译的工具是临时的，而且在最后会被丢弃，我们不推荐中等水平的读者运行第五章中的测试程序，这些测试程序的指令是为开发和测试人员准备的，它们不是必需的。

Binutils 和 GCC 的测试程序一个常见的问题就是在伪终端(PTY)外运行，这样可能会出现大量的测试失败。可能的原因有几个，但最可能的原因是宿主系统没有正确的设置 `devpts` 文件系统，这个问题在第五章中有详细的讨论。

某些软件包测试程序可能会给出错误的失败信息，请参考 LFS Wiki *http://www.linuxfromscratch.org/lfs/build-logs/6.2/* 以确认您所遇到的测试失败是否是预期会出现的。这个站点的内容对于本书所有的测试都是正确的。

5. 构建临时编译环境

5.1. 简介

本章介绍如何编译和安装一个小的 Linux 系统。这个系统将仅包含必要的工具，用于构建第六章中最终的 LFS 系统。

构建这个小系统分两步进行，第一步是构建一个新的不依赖于宿主系统的工具链(编译器、汇编器、连接器、库文件以及一些有用的软件)，第二个步骤是利用这个工具链去构建其它基本的工具。

本章中编译的文件将安装在 `$LFS/tools` 目录下，这样可以与下一章将要安装的软件以及宿主系统区分开来。这些软件包编译出来是起临时作用的，我们不希望这些软件和即将建立的 LFS 系统混杂在一起。

重要

在运行每一个软件包的编译指令之前，都需要用 `lfs` 用户解开这个软件包，并用 **cd** 命令进入软件包解开后的目录。编译指令假定您使用的是 **bash** shell 。

〔译者注〕举例来说，对于第一个软件包 Binutils-2.16.1 ，在执行书上的第一个命令

```
mkdir -v ../binutils-build
```

之前，必须先执行下列解包和切换目录的命令：

```
tar -xvjf $LFS/sources/binutils-2.16.1.tar.bz2 -C $LFS/sources/
cd $LFS/sources/binutils-2.16.1/
```

其他软件包以此类推。

某些软件包在编译之前需要打上补丁，但仅仅是需要补丁来解决某个问题的时候。一个补丁可能本章和下一章都会用到，也可能只在其中一章用到。因此，当某个软件包存在一个补丁，而在编译时并未让你使用这个补丁时，不要以为是我们忘记了。事实上，那个补丁只需要在另外一次编译中被用到。而应用某个补丁的时候，也可能会出现某些关于 *offset* 或 *fuzz* 的警告信息，不要理会，这个补丁仍然会被成功的应用。

大多数软件包的编译过程中，屏幕上可能会滚过一些警告信息，这些是正常且可以被安全忽略的。这些警告就像显示的那样：警告的是不标准，却不是不正确的 C 或 C++ 语法。C 标准常常会变，而某些软件包仍然使用老的标准，这不是问题，仅仅导致一些警告信息而已。

重要

在安装完每个软件包之后，删除其源代码和编译目录，除非另有特别说明。删除源文件可以节省磁盘空间，并且可以在下次需要安装同一个软件包的时候不会出现配置错误。

〔译者注〕举例来说，对于第一个软件包 Binutils-2.16.1 ，在执行完书上的最后一个命令

```
cp -v ld/ld-new /tools/bin
```

之后，可以使用下列命令删除其源文件和编译目录：

```
rm -fr $LFS/sources/binutils-2.16.1/
rm -fr $LFS/sources/binutils-build/
```

其他的软件包以此类推。

最后一次检查 `LFS` 环境变量是否设置正确：

```
echo $LFS
```

请确认输出显示的是挂载 LFS 分区的路径，在我们的例子中是 `/mnt/lfs` 。

5.2. 工具链技术说明

本节阐述了整个构建方法的一些基本原理和技术细节，您不必马上就理解本节中的所有内容。在您实际操作之后，就会了解大多数的东西，您可以在任何时候回顾本节。

第五章的总体目标是提供一个临时环境，您可以 chroot 到这个环境，在里面构建一个第六章中的干净、没有问题的目标 LFS 系统。为了尽量的与宿主系统分开，我们创建了一个自包含、自依赖的工具链。要注意的是，这个创建过程被设计为尽量减少新手犯错误的可能，同时尽可能多的提供教育价值。

重要

在继续之前，要先知道工作平台的名称，就是所谓的"target triplet"(目标三元组)，许多时候，target triplet 可能是 *i686-pc-linux-gnu* 。要确定 target triplet 的名称，有一个简单的方法就是运行许多源码包里都有的 `config.guess` 脚本。解开 Binutils 的源码包，然后运行脚本 **`./config.guess`** 并注意输出的内容。

工作平台的动态连接器名称也需要知道，这里指的是动态加载器(不要与 Binutils 里的标准连接器 `ld` 混淆了)。动态连接器由 Glibc 提供，用来找到并加载一个程序运行时所需的共享库，在做好程序运行的准备之后，运行这个程序。动态连接器的名称通常是 `ld-linux.so.2` ，在不怎么流行的平台上则可能是 `ld.so.1` ，而在新的 64 位平台上更可能是别的完全不同的名称。查看宿主系统的 `/lib` 目录可以确定动态连接器的名称。确定这个名称还有一个必杀技，就是在宿主系统上随便找一个二进制文件，运行 **`readelf -l <二进制文件名> | grep interpreter`** 并查看输出的内容。涵盖所有平台的权威参考请查看 Glibc 源码根目录里的 `shlib-versions` 文件。

第五章中构建方法是如何工作的一些技术要点：

- 这个过程在原理上与交叉编译类似，通过把工具安装在同一个目录(使用相同的"prefix")中以便协同工作，还利用了一点 GNU 的"魔法"。
- 小心处理标准连接器的库文件搜索路径，确保程序仅连接到指定的库上。
- 小心处理 `gcc` 的 `specs` 文件，告诉编译器要使用哪个动态连接器。

首先安装的是 Binutils ，因为 GCC 和 Glibc 的 `configure` 脚本要在汇编器和连接器上执行各种各样的特性测试，以确定软件的哪些功能要启用，哪些功能要禁用。这样做比你想像的还要重要，配置不正确的 GCC 或者 Glibc 会导致工具链出现微妙的错误，这样的错误造成的影响可能直到整个系统快要编译完成的时候才显现出来。测试程序通常会在其它的许多工作进行之前给出错误警告(以避免其后的无效劳动)。

Binutils 的汇编器和连接器安装在两个位置： `/tools/bin` 和 `/tools/$TARGET_TRIPLET/bin` ，一个位置的程序是另外一个位置的硬链接。连接器的一个重要方面是它的库搜索顺序，将 `--verbose` 选项传递给 `ld` 可以获得详细的信息。例如，输入命令： `ld --verbose | grep SEARCH` 将显示当前搜索路径和顺序。要显示 `ld` 连接的是哪些文件，可以编译一个伪(dummy)程序并把 `--verbose` 选项传递给连接器。举个例子，输入 `gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded` 将显示所有连接成功的文件。

第二个安装的软件包是 GCC 。下面是运行 `configure` 脚本时，输出内容的一个示例：

```
checking what assembler to use...
        /tools/i686-pc-linux-gnu/bin/as
checking what linker to use... /tools/i686-pc-linux-gnu/bin/ld
```

基于上面提到过的原因，这是重要的步骤，它同时证明了 GCC 的配置脚本并不是搜索 PATH 里的目录来寻找要使用哪个工具的，而且，在 `gcc` 的实际操作中，相同的搜索路径不一定会被使用。要知道 `gcc` 会使用哪个标准连接器，请运行 `gcc -print-prog-name=ld` 命令。

在编译一个伪程序的时候，给 `gcc` 命令传递 `-v` 选项可以获得详细的信息。举个例子： `gcc -v dummy.c` 将显示在预处理、编译和汇编各个阶段的详细信息，包括 `gcc` 文件包含的搜索路径及其顺序。

接下来安装的软件包是 Glibc 。编译 Glibc 的时候，最需要注意的地方是编译器、二进制工具(Binutils)和内核头文件。编译器一般不是问题，因为 Glibc 总是使用在 `PATH` 目录里找到的 `gcc` 。二进制工具和内核头文件则有点复杂，因此，为慎重起见，明确使用配置开关(选项)来强制进行正确的选择。在运行 `configure` 之后，请检查 `config.make` 文件的内容(位于 `glibc-build` 目录下)，查看所有重要的细节。注意， `CC="gcc -B/tools/bin/"` 的作用是控制要使用哪个二进制工具；而 `-nostdinc` 和 `-isystem` 选项则是控制编译器的文件包含搜索路径。这些选项表明了 Glibc 软件包的一个重要特征：根据其编译方法，它是非常自给自足的，而且一般不依赖于工具链的默认值。

装完 Glibc 之后，需要做一些调整使得只在 `/tools` 目录里搜索和连接。安装一个调整好的 `ld` ，将它的固化搜索路径限制在 `/tools/lib` 目录。然后修改 `gcc` 的 specs 文件以指向 `/tools/lib` 目录里新的动态连接器。最后这一步在整个过程中至关重要，像上面提到的，指向动态连接器的固化路径被嵌入到每个 ELF 可执行文件里。可以通过 `readelf -l <二进制文件名> | grep interpreter` 命令来检查。修改 gcc 的 specs 文件以确保本章后面编译的每一个程序都使用位于 `/tools/lib` 目录里新的动态连接器。

需要使用新动态连接器也是第二遍编译 GCC 需要打 Specs 补丁的原因。不这样做的结果是 GCC 会把宿主系统 `/lib` 目录下动态连接器的名字嵌入进来，这样有悖于与宿主系统隔离的目标。

第二遍编译 Binutils 的过程中，我们利用 `--with-lib-path` 选项来控制 `ld` 的库搜索路径。如前面所指出的，核心工具链是自包含和自依赖的，所以第五章余下的软件包的编译将依赖于 `/tools` 下新的 Glibc 。

在第六章进入虚根环境后，第一个安装的主要软件包就是 Glibc ，原因是上面所提到的 Glibc 自给自足的特性。一旦 Glibc 安装到 `/usr` 目录后，马上改变工具链的默认值，然后构建目标 LFS 系统的其它部分。

5.3. Binutils-2.16.1 - 第一遍

Binutils 是一组开发工具，包括连接器、汇编器和其他用于目标文件和档案的工具。

预计编译时间： 1 SBU所需磁盘空间： 189 MB

5.3.1. 安装 Binutils

首先安装的第一个软件包是 Binutils ，这非常重要，因为 Glibc 和 GCC 会针对可用的连接器和汇编器进行多种测试，以决定是否打开某些特性。

Binutils 的文档推荐用一个新建的目录来编译它，而不是在源码目录中：

```
mkdir -v ../binutils-build
cd ../binutils-build
```

注意

如果你想使用本书余下部份列出的 SBU 值，那么现在就要测量一下编译本软件包的时间。你可以用类似下面这样的 `time` 命令： **`time { ./configure ... && make && make install; }`** 。

为编译 Binutils 做准备：

```
../binutils-2.16.1/configure --prefix=/tools --disable-nls
```

配置选项的含义：

`--prefix=/tools`

这个参数告诉 configure 脚本，应该把 Binutils 软件包中的程序安装到 `/tools` 目录中。

`--disable-nls`

这个参数禁止了国际化(通常简称i18n)，静态程序不需要国际化的特性。

接下来编译它：

```
make
```

现在编译完成了。通常我们会运行测试套件，但是目前测试套件(Tcl, Expect, DejaGNU)尚未安装。而且在这里运行测试也没什么用处，因为第一遍安装的程序很快就会被第二遍的程序所覆盖。

安装软件包：

```
make install
```

接下来为后面"调整工具链"步骤准备连接器：

```
make -C ld clean
make -C ld LIB_PATH=/tools/lib
cp -v ld/ld-new /tools/bin
```

make 参数的含义：

`-C ld clean`

告诉 make 程序删除所有 `ld` 子目录中编译生成的文件。

`-C ld LIB_PATH=/tools/lib`

这个选项重新编译 `ld` 子目录中的所有文件。在命令行中指定 Makefile 的 `LIB_PATH` 变量值，使它明确指向临时工具目录，以覆盖默认值。这个变量的值指定了连接器的默认库搜索路径，它在这一章的稍后部分会用到。

关于这个软件包的详细资料位于节 6.11.2, Binutils 的内容

5.4. GCC-4.0.3 - 第一遍

GCC 软件包包含 GNU 编译器集合，其中有 C 和 C++ 编译器。

预计编译时间： 8.2 SBU所需磁盘空间： 514 MB

5.4.1. 安装 GCC

GCC 的安装指南推荐用一个新建的目录来编译它，而不是在源码目录中：

```
mkdir -v ../gcc-build
cd ../gcc-build
```

为编译 GCC 做准备：

```
../gcc-4.0.3/configure --prefix=/tools \
    --with-local-prefix=/tools --disable-nls --enable-shared \
    --enable-languages=c
```

配置选项的含义：

`--with-local-prefix=/tools`

这个参数的目的是把 `/usr/local/include` 目录从 `gcc` 的 include 搜索路径里删除。这并不是绝对必要，但我们想尽量减小宿主系统的影响，所以才这样做。

`--enable-shared`

这个参数咋一看有点违反直觉。但只有加上它，才能编译出 `libgcc_s.so.1` 和 `libgcc_eh.a` 。Glibc(下一个软件包)的配置脚本只有在找到 `libgcc_eh.a` 时才能确保产生正确的结果。

`--enable-languages=c`

只编译 GCC 软件包中的 C 编译器。我们在本章里不需要其它编译器。

接下来编译它：

```
make bootstrap
```

make 参数的含义：

`bootstrap`

使用这个参数的目的不仅仅是编译 GCC ，而是重复编译它几次。它用第一次编译生成的程序来第二次编译自己，然后又用第二次编译生成的程序来第三次编译自己，最后比较第二次和第三次编译的结果，以确保编译器能毫无差错的编译自身，这通常表明编译是正确的。

编译现在完成了，通常我们会在这里运行测试套件，但是正如前面说过的，测试套件目前尚未安装，而且在这里运行测试没什么用处，因为第一遍安装的程序很快就会被第二遍的程序所覆盖。

安装软件包：

```
make install
```

最后，我们创建一个必要的符号连接。因为许多程序和脚本试图运行 `cc` 而不是 `gcc` ，这样做是为了让程序能在多种 Unix 平台上运行，并保持一致性。并不是每个人都安装 GNU C 编译器。只运行 `cc` 而不是 `gcc` 可以把选择 C 编译器的自由留给系统管理员，我们这里将指向 `gcc` ：

```
ln -vs gcc /tools/bin/cc
```

关于这个软件包的详细资料位于节 6.12.2, GCC 的内容

5.5. Linux-Libc-Headers-2.6.12.0

Linux-Libc-Headers 包含了"纯净的"内核头文件。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 27 MB

5.5.1. 安装 Linux-Libc-Headers

多年来的公共惯例是使用 `/usr/include` 目录下"原始的"内核头文件(直接来自于内核源码包)，但是近年来，内核开发者强烈要求不要这样做，因此诞生了 Linux-Libc-Headers 项目，其目标是维护一个API(应用程序编程接口)版本稳定的 Linux 头文件。

安装这些头文件：

```
cp -Rv include/asm-i386 /tools/include/asm
cp -Rv include/linux /tools/include
```

如果您的机器不是 i386 兼容架构的，请相应的调整第一条命令。

关于这个软件包的详细资料位于节 6.7.2, Linux-Libc-Headers 的内容

5.6. Glibc-2.3.6

Glibc 包含了主要的C库。这个库提供了基本例程，用于分配内存、搜索目录、打开关闭文件、读写文件、字串处理、模式匹配、数学计算等等。

预计编译时间： 6 SBU所需磁盘空间： 325 MB

5.6.1. 安装 Glibc

Glibc 文档推荐在源码目录之外的一个专门的编译目录下进行编译：

```
mkdir -v ../glibc-build
cd ../glibc-build
```

接下来为编译 Glibc 做准备：

```
../glibc-2.3.6/configure --prefix=/tools \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --with-binutils=/tools/bin \
    --without-gd --with-headers=/tools/include \
    --without-selinux
```

配置选项的含义：

--disable-profile

它关掉了 profiling 信息相关的库文件编译。如果你打算做 profiling ，就省掉这个参数。

--enable-add-ons

这个指示 Glibc 使用附加的 NPTL 包作为线程库。

--enable-kernel=2.6.0

这个告诉 Glibc 编译支持 2.6.x 内核的库。

--with-binutils=/tools/bin

这个参数并不是必需的。但它们能保证在编译 Glibc 时不会用错 Binutils 程序。

--without-gd

这个参数保证不生成 `memusagestat` 程序，这个程序会顽固地连接到宿主系统的库文件(libgd, libpng, libz 等等)。

--with-headers=/tools/include

这个参数指示 Glibc 按照前面刚刚安装到 tools 目录中的内核头文件编译自己，从而精确的知道内核的特性以根据这些特性对自己进行最佳化编译。

`--without-selinux`

当从一个含有 SELinux 特性的宿主系统(如 Fedora Core 3)编译时，Glibc 将会将 SELinux 支持编译进来。由于 LFS 工具链并不包含 SELinux 支持，所以一个含有 SELinux 特性的 Glibc 将会导致许多操作失败。所以这里明确禁用它。

在这个阶段你可能会看到下面的警告：

```
configure: WARNING:
*** These auxiliary programs are missing or
*** incompatible versions: msgfmt
*** some features will be disabled.
*** Check the INSTALL file for required versions.
```

抱怨说缺少或有不兼容的 `msgfmt` 程序，这没有什么大问题，不过有时候可能会在运行测试套件的时候出问题。 `msgfmt` 程序是宿主系统 Gettext 应当提供的一部分。如果担心宿主系统的 `msgfmt` 有兼容性问题，你可以升级宿主系统的 Gettext ，也可以忽略这个问题而不去管它。

编译软件包：

```
make
```

现在编译完成了。正如前面说过的，在这里运行测试没什么用处，但是如果你坚持要测试的话可以运行下列命令：

```
make check
```

关于测试失败重要性的讨论请参考这里：节 6.9, "Glibc-2.3.6."

在这一章里，Glibc 的测试套件高度依赖于宿主系统的工具和环境，尤其是内核。因为这个原因，有时错误很难避免，但是无需太在意。第六章里面的 Glibc 才是我们最后所使用的，那里的 Glibc 需要通过绝大多数测试。但要注意的是，即使在第六章里，有的失败还是会出现，比如 math 测试。

当遇到一个错误时，记录下来，再用 `make check` 继续。测试套件会从出错的地方继续进行。你也可以用 `make -k check` 来一次把测试做完。但如果你这样做的话，就要把屏幕输出记录到文件里（ `make -k check > ck_log` ），以便最后检查到底出了多少错，哪些测试出错了。

在安装 Glibc 的过程中，它会警告缺少 `/tools/etc/ld.so.conf` 文件。其实这没什么关系，不过下面的命令能修正它：

```
mkdir -v /tools/etc
touch /tools/etc/ld.so.conf
```

安装软件包：

```
make install
```

不同的国家和文化，使用不同的习俗来交流。这样的习俗很多，从比较简单的时间和日期格式，到非常复杂的语言发音。GNU 程序的"internationalization"(国际化，又称"i18n"，18表示中间的18个字母)是以 locale 来实现的。

注意

如果刚才没有运行测试套件，那么现在就没有必要安装 locale 。在下一章里面我们将会安装。如果你一定要安装 locale ，请参考节 6.9, "Glibc-2.3.6."的内容。

关于这个软件包的详细资料位于节 6.9.4, Glibc 的内容

5.7. 调整工具链

现在临时的C库已经装好，接下来本章中要编译的所有工具应该连接到这些库上。为了达到这个目标，需要调整连接器和编译器的 specs 文件。

在第一遍编译 Binutils 快结束时已经调整过的连接器，现在需要被重新命名以便可以被正确的找到和使用。首先备份原来的连接器，然后用调整过的连接器来替代，最后还要创建一个指向 `/tools/$(gcc -dumpmachine)/bin` 中连接器副本的连接。

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

从现在开始，所有程序都将连接到 `/tools/lib` 中的库文件。

下面要做的是修正 GCC 的"specs"文件，使它指向新的动态连接器。一个简单的 `sed` 命令就能做到：

```
SPECFILE=`dirname $(gcc -print-libgcc-file-name)`/specs &&
gcc -dumpspecs > $SPECFILE &&
sed 's@^/lib/ld-linux.so.2@/tools&@g' $SPECFILE > tempspecfile &&
mv -vf tempspecfile $SPECFILE &&
unset SPECFILE
```

推荐你拷贝和粘贴上面的命令，而不是手动输入。当然你也可以手动编辑 specs 文件，只要把所有的"/lib/ld-linux.so.2"都替换成"/tools/lib/ld-linux.so.2"就行了。

请用你的眼睛亲自仔细检查一下 specs 文件，以确保上述修改的的确确生效了。

重要

如果你的系统平台上，动态连接器的名字不是 `ld-linux.so.2` ，你必须把上面命令里的"ld-linux.so.2"换成你的系统平台上动态连接器的名字。参见节 5.2, "工具链技术说明,"。

在编译过程中，GCC 会运行 `fixincludes` 脚本来扫描系统头文件目录，并找出需要修正的头文件(比如包含语法错误)，然后把修正后的文件放到 GCC 专属头文件目录里。因此，它可能会找出宿主系统中需要修正的头文件，并将修正后的结果放到 GCC 专属头文件目录里。由于本章的剩余部分仅需要使用当前已经安装好的 GCC 和 Glibc 的头文件，所以任何"修正后的"头文件都可以被安全的删除。并且这样做也有助于避免宿主系统中的头文件"污染"编译环境。运行下面的命令删除 GCC 专属头文件目录中的头文件(由于命令较长，推荐你拷贝和粘贴命令，而不是手动输入)：

```
GCC_INCLUDEDIR=`dirname $(gcc -print-libgcc-file-name)`/include &&
find ${GCC_INCLUDEDIR}/* -maxdepth 0 -xtype d -exec rm -rvf '{}' \; &&
rm -vf `grep -l "DO NOT EDIT THIS FILE" ${GCC_INCLUDEDIR}/*` &&
unset GCC_INCLUDEDIR
```

小心

现在，需要停下来确认新工具链的基本功能(编译和连接)是否按预期工作，运行下面的命令做一个简单的合理性检查：

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

如果一切正常，应该不会出错，而且最后一个命令的结果应当是：

```
[Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

注意， `/tools/lib` 应该是动态连接器的前缀。

如果输出不是像上面那样或者根本没有输出，那么就有大问题了。返回并检查前面的操作，找出问题，并改正过来。在改正之前，不要继续后面的部份，因为这样做没有意义。首先，再次上述合理性检查，用 `gcc` 代替 `cc` ，如果工作正常，那么是因为 `/tools/bin/cc` 这个符号链接丢失了。回头看看节 5.4, "GCC-4.0.3 - 第一遍,"，并建立符号链接。接下来，确保 `PATH` 正确。检查时，运行 `echo $PATH` 并检查 `/tools/bin` 是否在列表的最前面。如果 `PATH` 错误，可能是因为你没有以 `lfs` 用户登录，或者在节 4.4, "设置工作环境."部分出错了。另外一个原因可能是上面修正 specs 文件时出错，如果这样，重新修改 specs 文件，复制粘贴时要小心仔细。

在确定一切正常后，删除测试文件：

```
rm -v dummy.c a.out
```

注意

下一小节中编译 TCL 时也将有助于检查工具连是否正确。如果 TCL 编译失败则表示之前安装的 Binutils 、GCC 或 Glibc 有问题，而不是 TCL 自身有问题。

5.8. Tcl-8.4.13

Tcl 软件包包含工具命令语言(Tool Command Language)。

预计编译时间： 0.3 SBU所需磁盘空间： 24 MB

5.8.1. 安装 Tcl

这个软件包和接下来安装的两个软件包(Expect 和 DejaGNU)是为了给运行 GCC 和 Binutils 的测试程序提供支持。仅为了测试而安装三个软件包，看起来似乎有点多余，但是看到那些最重要的工具正常工作，心理上会比较踏实。即使没有运行本章中测试程序(不是必需的)，运行第六章中的测试时也需要这些软件包。

为编译 Tcl 做准备：

```
cd unix
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： **`TZ=UTC make test`** 。已知 Tcl 的测试程序会在某些还未完全了解的宿主系统下出现测试失败的情况，因此，如果这里的测试失败了，不要紧，因为这并不关键。 *`TZ=UTC`* 参数将时区设置为协调世界时(UTC)，也就是格林尼治时间(GMT)，但只是在运行测试程序的时候才这样设置，这将确保时钟测试正确。关于 `TZ` 环境变量的详细资料位于第七章。

安装软件包：

```
make install
```

安装 Tcl 头文件，下一个包(Expect)要使用 Tcl 的头文件。

```
make install-private-headers
```

现在创建一个必需的符号链接：

```
ln -sv tclsh8.4 /tools/bin/tclsh
```

5.8.2. Tcl 的内容

安装的程序： tclsh(→tclsh8.4), tclsh8.4安装的库： libtcl8.4.so

简要描述

`tclsh8.4`	Tcl 命令 shell
`tclsh`	指向 tclsh8.4 的链接
`libtcl8.4.so`	Tcl 库文件

5.9. Expect-5.43.0

Expect 软件包包含一个通过执行脚本对话框与其它交互式程序通信的工具。

预计编译时间： 0.1 SBU所需磁盘空间： 4 MB

5.9.1. 安装 Expect

先修正一个可能导致 GCC 测试程序假失败的 bug ：

```
patch -Np1 -i ../expect-5.43.0-spawn-1.patch
```

为编译 Expect 做准备：

```
./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=/tools/include --with-x=no
```

配置选项的含义：

`--with-tcl=/tools/lib`

这个选项确保配置脚本找到的是安装在临时工具目录下的 Tcl ，而不是宿主系统里的。

`--with-tclinclude=/tools/include`

这个选项告诉 Expect 到哪里寻找 Tcl 的源代码目录和头文件。使用这个选项可以避免 `configure` 脚本因为找不到 Tcl 的源代码目录而导致的失败。

`--with-x=no`

这个选项告诉 `configure` 脚本不要搜索 Tk(Tcl的图形界面组件)或者 X Window 系统的库，这两者都可能位于宿主系统上。

编译软件包：

```
make
```

要测试结果，请运行： **`make test`** 。请注意，已知 Expect 的测试程序会在某些不在我们控制范围内的宿主系统下出现测试失败。因此，如果您运行这里的测试程序失败了也没关系，因为这并不关键。

安装软件包：

```
make SCRIPTS="" install
```

make 参数的含义：

`SCRIPTS=""`

这个选项防止安装 Expect 所补充的一些并不需要的脚本。

5.9.2. Expect 的内容

安装的程序： expect安装的库： libexpect-5.43.a

简要描述

`expect`	按照一个脚本与其它交互式程序通信
`libexpect-5.43.a`	包含的函数可以让 Expect 作为 Tcl 的扩展来使用，或者直接被 C 或 C++ 使用(不需要 Tcl)

5.10. DejaGNU-1.4.4

DejaGNU 软件包包含了一个测试其它程序的框架。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 6.2 MB

5.10.1. 安装 DejaGNU

为编译 DejaGNU 做准备：

```
./configure --prefix=/tools
```

编译并安装软件包：

```
make install
```

要测试结果，请运行： **`make check`** 。

5.10.2. DejaGNU 的内容

安装的程序： runtest

简要描述

`runtest`	一个包装脚本，用于定位正确的 `expect` 解释程序(shell)并运行 DejaGNU

5.11. GCC-4.0.3 - 第二遍

GCC 软件包包含 GNU 编译器集合，其中有C和C++编译器。

预计编译时间： 4.2 SBU所需磁盘空间： 443 MB

5.11.1. 重新安装 GCC

测试 GCC 和 Binutils 所需的工具(Tcl, Expect, DejaGNU)已经安装好。现在 GCC 和 Binutils 将被重新编译，连接到新的 Glibc 并作适当测试(如果运行这章中的测试的话)。注意，这些测试套件受伪终端(PTY)的影响很大，这些伪终端通常是由宿主系统通过 `devpts` 文件系统实现的。你可以用下面的方法，来测试宿主系统中PTY是否设置正常：

```
expect -c "spawn ls"
```

如果你得到下面的回答：

```
The system has no more ptys.
Ask your system administrator to create more.
```

说明主系统的 PTY 没设置好。这种情况下，运行 GCC 和 Binutils 的测试套件就没什么意义了。你需要先解决主系统中的 PTY 设置问题。具体请参见 LFS 的 FAQ ：*http://www.linuxfromscratch.org/lfs/faq.html#no-ptys* 。

在之前的节 5.7, "调整工具链"中我们提到过在 GCC 编译过程中会运行 `fixincludes` 脚本来扫描系统头文件目录，并找出需要修正的头文件，然后把修正后的头文件放到 GCC 专属头文件目录里。因为现在 GCC 和 Glibc 已经安装完毕，而且它们的头文件已知无需修正，所以这里并不需要 `fixincludes` 脚本。另外，由于 GCC 专属头文件目录会被优先搜索，结果就是 GCC 使用的头文件是宿主系统的头文件，而不是新安装的那个，从而导致编译环境被"污染"。因此必须通过下面的命令来禁止 `fixincludes` 脚本运行：

```
cp -v gcc/Makefile.in{,.orig} &&
sed 's@\./fixinc\.sh@-c true@' gcc/Makefile.in.orig > gcc/Makefile.in
```

在节 5.4, "GCC-4.0.3 - 第一遍"中进行的 bootstrap 编译使用了 `-fomit-frame-pointer` 选项，而非 bootstrap 编译则默认忽略了该选项，所以需要使用下面的 `sed` 命令来确保在非 bootstrap 编译时也同样使用 `-fomit-frame-pointer` 选项，以保持一致性：

```
cp -v gcc/Makefile.in{,.tmp} &&
sed 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \
  > gcc/Makefile.in
```

使用下面的补丁修改 GCC 的缺省动态连接器(通常是 `ld-linux.so.2`)的位置：

```
patch -Np1 -i ../gcc-4.0.3-specs-1.patch
```

上述补丁还把 `/usr/include` 从 GCC 的头文件搜索路径里删掉。现在预先打补丁而不是在安装 GCC 之后调整 specs 文件可以保证新的动态连接器在编译 GCC 的时候就用上。也就是说，随后的所有临时程序都会连接到新的 Glibc 上。

重要

上述补丁非常重要，为了成功编译，千万别忘了使用它们。

再次为编译创建一个单独目录：

```
mkdir -v ../gcc-build
cd ../gcc-build
```

在开始编译前，别忘了 unset 任何优化相关的环境变量。[是不是应当省略这句?]

为编译 GCC 做准备：

```
../gcc-4.0.3/configure --prefix=/tools \
    --with-local-prefix=/tools --enable-clocale=gnu \
    --enable-shared --enable-threads=posix \
    --enable-__cxa_atexit --enable-languages=c,c++ \
    --disable-libstdcxx-pch
```

新配置选项的含义：

`--enable-clocale=gnu`

本参数确保 C++ 库在任何情况下都使用正确的 locale 模块。如果配置脚本查找到 *de_DE* 这个 locale ，它就会使用正确的 gnu locale 模块。然而，如果没有安装 *de_DE* ，就有可能创建出应用程序二进制接口(ABI)不兼容的 C++ 库文件，这是因为选择了错误的通用(generic) locale 模块。

`--enable-threads=posix`

使 C++ 异常能处理多线程代码。

`--enable-__cxa_atexit`

用 *__cxa_atexit* 代替 *atexit* 来登记 C++ 对象的本地静态和全局析构函数，这是为了完全符合标准对析构函数的处理规定。它还会影响到 C++ ABI ，因此生成的 C++ 共享库在其他的 Linux 发行版上也能使用。

`--enable-languages=c,c++`

本参数编译 C 和 C++ 语言的编译器。

--disable-libstdcxx-pch

不为 `libstdc++` 编译预编译头(PCH)，它占用了很大空间，但是我们用不到它。

编译软件包：

```
make
```

现在没必要用 *bootstrap* 作为 make 的目标，因为这里 GCC 是用相同版本的 GCC 来编译的，其实连源码都一模一样，就是在第一遍的时候安装的那个。

现在编译完成了，早先我们谈到过，本章中的临时工具的测试程序并不是必须运行的，如果您要运行 GCC 的测试程序，请输入下面的命令：

```
make -k check
```

-k 参数是让测试套件即使遇到错误，也继续运行，直到完成。GCC 的测试套件非常全面，所以基本上总是会出一些错的。

对于测试错误的重要说明位于节 6.12, "GCC-4.0.3."。

安装软件包：

```
make install
```

小心

现在，需要停下来再一次确认新工具链的基本功能(编译和连接)是否按预期工作，运行下面的命令做一个简单的合理性检查：

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

如果一切正常，应该不会出错，而且最后一个命令的结果应当是：

```
[Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

注意，`/tools/lib` 应该是动态连接器的前缀。

如果输出不是像上面那样或者根本没有输出，那么就有大问题了。返回并检查前面的操作，找出问题，并改正过来。在改正之前，不要继续后面的部份，因为这样做没有意义。首先，再次上述合理性检查，用 `gcc` 代替 `cc` ，如果工作正常，那么是因为 `/tools/bin/cc` 这个

符号链接丢失了。回头看看节 5.4, "GCC-4.0.3 - 第一遍," 并建立符号链接。接下来，确保 `PATH` 正确。检查时，运行 `echo $PATH` 并检查 `/tools/bin` 是否在列表的最前面。如果 `PATH` 错误，可能是因为你没有以 `lfs` 用户登录，或者在 节 4.4, "设置工作环境."部分出错了。另外一个原因可能是上面修正 specs 文件时出错，如果这样，重新修改 specs 文件，复制粘贴时小心。

在确定一切正常后，删除测试文件：

```
rm -v dummy.c a.out
```

关于这个软件包的详细资料位于节 6.12.2, GCC 的内容

5.12. Binutils-2.16.1 - 第二遍

Binutils 是一组开发工具，包括连接器、汇编器和其他用于目标文件和档案的工具。

预计编译时间： 1.1 SBU所需磁盘空间： 154 MB

5.12.1. 重新安装 Binutils

再次为编译创建一个单独目录：

```
mkdir -v ../binutils-build
cd ../binutils-build
```

为编译 Binutils 做准备：

```
../binutils-2.16.1/configure --prefix=/tools \
    --disable-nls --with-lib-path=/tools/lib
```

新配置选项的含义：

`--with-lib-path=/tools/lib`

这个选项指示 configure 脚本在 Binutils 编译过程中将传递给连接器的库搜索路径设为 `/tools/lib` ，以防止连接器搜索宿主系统的库目录。

编译软件包：

```
make
```

现在编译完成了，早先我们谈到过，本章中的临时工具的测试程序并不是必须运行的，如果您要运行 Binutils 的测试程序，请输入下面的命令：

```
make check
```

安装软件包：

```
make install
```

现在，为下一章的"再次调整工具链"阶段配置连接器：

```
make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
cp -v ld/ld-new /tools/bin
```

关于这个软件包的详细资料位于节 6.11.2, Binutils 的内容

5.13. Ncurses-5.5

Ncurses 提供独立于终端的字符终端处理库，含有功能键定义(快捷键)、屏幕绘制以及基于文本终端的图形互动功能。

预计编译时间： 0.7 SBU所需磁盘空间： 30 MB

5.13.1. 安装 Ncurses

为编译 Ncurses 做准备：

```
./configure --prefix=/tools --with-shared \
    --without-debug --without-ada --enable-overwrite
```

配置选项的含义：

`--without-ada`

这个选项让 Ncurses 在即使宿主系统上安装了 Ada 编译器的情况下也不要编译其 Ada 绑定。需要这样做的原因是一旦我们进入 `chroot` 环境，Ada 就不能使用了。

`--enable-overwrite`

这个选项让 Ncurses 把它的头文件安装到 `/tools/include` 目录，而不是 `/tools/include/ncurses` 目录，以确保其它软件包可以顺利找到 Ncurses 的头文件。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.18.2, Ncurses 的内容

5.14. Bash-3.1

Bash 是 Bourne-Again Shell 的缩写，它在 UNIX 系统中作为 shell 被广泛使用。

预计编译时间： 0.4 SBU所需磁盘空间： 22 MB

5.14.1. 安装 Bash

Bash-3.1 正式发布以后开发者们又修正了许多缺陷，下面的补丁包含了这些修正：

```
patch -Np1 -i ../bash-3.1-fixes-8.patch
```

为编译 Bash 做准备：

```
./configure --prefix=/tools --without-bash-malloc
```

配置选项的含义：

`--without-bash-malloc`

这个选项禁用了 Bash 的内存分配函数(`malloc`)，这个函数已知会造成段错误，通过设置这个选项，Bash 将使用更为稳定的 Glibc 里的 `malloc` 函数。

编译软件包：

```
make
```

要测试结果，请运行： **`make tests`** 。

安装软件包：

```
make install
```

为那些用 `sh` 作为 shell 的程序创建符号链接：

```
ln -vs bash /tools/bin/sh
```

关于这个软件包的详细资料位于节 6.27.2, Bash 的内容

5.15. Bzip2-1.0.3

Bzip2 包含了对文件进行压缩和解压缩的工具，对于文本文件，`bzip2` 比传统的 `gzip` 拥有更高压缩比。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 4.2 MB

5.15.1. 安装 Bzip2

Bzip2 软件包没有 `configure` 脚本，用下面的命令直接编译：

```
make
```

安装软件包：

```
make PREFIX=/tools install
```

关于这个软件包的详细资料位于节 6.28.2, Bzip2 的内容

5.16. Coreutils-5.96

Coreutils 软件包包括一整套用于显示和设置基本系统特征的工具。

预计编译时间： 0.6 SBU所需磁盘空间： 56.1 MB

5.16.1. 安装 Coreutils

为编译 Coreutils 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： **`make RUN_EXPENSIVE_TESTS=yes check`** ，*RUN_EXPENSIVE_TESTS=yes* 参数让测试程序运行几个附加的测试，在某些平台上这些测试会耗费更多的 CPU 和内存，不过一般在 Linux 上不是什么问题。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.14.2, Coreutils 的内容

5.17. Diffutils-2.8.1

Diffutils 软件包里的程序可以向你显示两个文件或目录的差异，常用来生成软件的补丁。

预计编译时间： 0.1 SBU所需磁盘空间： 6.2 MB

5.17.1. 安装 Diffutils

为编译 Diffutils 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.29.2, Diffutils 的内容

5.18. Findutils-4.2.27

Findutils 软件包包含查找文件的工具，既能即时查找(递归的搜索目录，并可以显示、创建和维护文件)，也能在数据库里查找(通常比递归查找快但是在数据库没有及时更新的情况下，结果并不可靠)。

预计编译时间： 0.2 SBU所需磁盘空间： 12 MB

5.18.1. 安装 Findutils

为编译 Findutils 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.32.2, Findutils 的内容

5.19. Gawk-3.1.5

Gawk 是一个处理文本文件的工具包。

预计编译时间： 0.2 SBU所需磁盘空间： 18.2 MB

5.19.1. 安装 Gawk

为编译 Gawk 做准备：

```
./configure --prefix=/tools
```

由于 `configure` 脚本的一个 bug ，Gawk 不能正确检测某些 Glibc 支持的 locale ，这将会导致一些问题，比如，Gettext 的测试程序会失败。修复这个 bug 的办法是在 `config.h` 文件结尾追加丢失的宏定义：

```
cat >>config.h <<"EOF"
#define HAVE_LANGINFO_CODESET 1
#define HAVE_LC_MESSAGES 1
EOF
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.35.2, Gawk 的内容

5.20. Gettext-0.14.5

Gettext 包含用于系统的国际化和本地化的工具，可以在编译程序的时候使用本国语言支持(NLS)，可以使程序的输出使用用户设置的语言而不是英文。

预计编译时间： 0.4 SBU所需磁盘空间： 43 MB

5.20.1. 安装 Gettext

对于临时工具链来说，我们只需要编译和安装 Gettext 中的一个二进制文件即可。

为编译 Gettext 做准备：

```
cd gettext-tools
./configure --prefix=/tools --disable-shared
```

配置选项的含义：

`--disable-shared`

当前我们不需要安装任何 Gettext 共享库，因此也就不需要编译它们。

编译软件包：

```
make -C lib
make -C src msgfmt
```

因为只编译了一个二进制文件，所以无法运行测试套件。并且我们也不推荐在此时运行测试。

安装编译好的二进制文件 `msgfmt` ：

```
cp -v src/msgfmt /tools/bin
```

关于这个软件包的详细资料位于节 6.36.2, Gettext 的内容

5.21. Grep-2.5.1a

Grep 可以按指定的匹配模式搜索文件中的内容。

预计编译时间： 0.1 SBU所需磁盘空间： 4.8 MB

5.21.1. 安装 Grep

为编译 Grep 做准备：

```
./configure --prefix=/tools \
    --disable-perl-regexp
```

配置选项的含义：

`--disable-perl-regexp`

这个选项确保 `grep` 程序不连接可能在宿主系统上存在的 PCRE(Perl 兼容正则表达式)库，因为进入 `chroot` 环境后，它就不能使用了。

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.37.2, Grep 的内容

5.22. Gzip-1.3.5

Gzip 软件包包含了压缩和解压文件的程序。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 2.2 MB

5.22.1. 安装 Gzip

为编译 Gzip 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.39.2, Gzip 的内容

5.23. M4-1.4.4

M4 包含一个宏处理器。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 3 MB

5.23.1. 安装 M4

为编译 M4 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.16.2, M4 的内容

5.24. Make-3.80

Make 软件包包含一个编译软件包的程序。

预计编译时间： 0.1 SBU所需磁盘空间： 7.8 MB

5.24.1. 安装 Make

为编译 Make 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.44.2, Make 的内容

5.25. Patch-2.5.4

Patch 软件包包含一个根据补丁文件来修改原文件的程序。补丁文件通常是用 `diff` 程序创建的。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 1.6 MB

5.25.1. 安装 Patch

为编译 Patch 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.48.2, Patch 的内容

5.26. Perl-5.8.8

Perl 是 Practical Extraction and Report Language的缩写。Perl 将 C, sed, awk 和 sh 的最佳特性集于一身，是一种强大的编程语言。

预计编译时间： 0.7 SBU所需磁盘空间： 84 MB

5.26.1. 安装 Perl

首先应用下面的补丁，调整指向 C 库的硬连接路径：

```
patch -Np1 -i ../perl-5.8.8-libc-2.patch
```

准备编译 Perl(请正确输入下面命令中的'Data/Dumper Fcntl IO POSIX'——全是字母)：

```
./configure.gnu --prefix=/tools -Dstatic_ext='Data/Dumper Fcntl IO POSIX'
```

配置选项的含义：

-Dstatic_ext='Data/Dumper Fcntl IO POSIX'

这个选项让 Perl 编译静态扩展的最小集，下一章安装和测试 Coreutils 软件包的时候需要用到。

仅需要编译这个软件包中的一小部分必要工具：

```
make perl utilities
```

尽管 Perl 附带测试程序，但我们不推荐在这里运行。由于只编译了一部分 Perl，现在运行 `make test` 会编译 Perl 的其余部分，而这里我们并不需要它们。如果想测试的话，可以到下一章再运行测试程序。

安装这些工具和库：

```
cp -v perl pod/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.8.8
cp -Rv lib/* /tools/lib/perl5/5.8.8
```

关于这个软件包的详细资料位于节 6.22.2, Perl 的内容

5.27. Sed-4.1.5

sed 是一个流编辑程序，在一个输入流(从一个文件或者一个管道的输入)上进行基本的文本编辑操作。

预计编译时间： 0.1 SBU所需磁盘空间： 6.1 MB

5.27.1. 安装 Sed

为编译 Sed 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.20.2, Sed 的内容

5.28. Tar-1.15.1

Tar 软件包含有一个归档程序，用来保存文件到归档文件或者从给定的 tar 归档文件中释放文件。

预计编译时间： 0.2 SBU所需磁盘空间： 13.7 MB

5.28.1. 安装 Tar

如果想运行测试套件则需要使用下面的补丁修正一些与 GCC-4.0.3 相关的问题：

```
patch -Np1 -i ../tar-1.15.1-gcc4_fix_tests-1.patch
```

为编译 Tar 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.53.2, Tar 的内容

5.29. Texinfo-4.8

Texinfo 软件包包含读取、写入和转换 Info 文档的程序，以提供系统文档。

预计编译时间： 0.2 SBU所需磁盘空间： 16.3 MB

5.29.1. 安装 Texinfo

为编译 Texinfo 做准备：

```
./configure --prefix=/tools
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

关于这个软件包的详细资料位于节 6.54.2, Texinfo 的内容

5.30. Util-linux-2.12r

Util-linux 软件包包含许多工具。其中比较重要的是加载、卸载、格式化、分区和管理驱动器，以及打开 tty 端口和处理消息。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 8.9 MB

5.30.1. 安装 Util-linux

Util-linux 默认不使用刚才安装在 `/tools` 目录下的头文件和库文件，我们更改配置脚本来修正这个问题：

```
sed -i 's@/usr/include@/tools/include@g' configure
```

为编译 Util-linux 做准备：

```
./configure
```

编译一些支持例程：

```
make -C lib
```

我们只需要这个软件包中的少数几个工具，因此只需要编译这几个工具就可以了：

```
make -C mount mount umount
make -C text-utils more
```

这个软件包没有附带测试程序。

把这些程序复制到临时工具目录：

```
cp mount/{,u}mount text-utils/more /tools/bin
```

关于这个软件包的详细资料位于节 6.56.3, Util-linux 的内容

5.31. 清理系统

本节的步骤是可选的，但如果 LFS 分区实在很小则除外；同时了解哪些东西是不必要的、可以删除的也是有好处的。到目前为止已经安装的可执行程序和库文件包含大约 70 MB 不必要的调试符号，运行下面的命令删除这些符号：

```
strip --strip-debug /tools/lib/*
strip --strip-unneeded /tools/{,s}bin/*
```

上面的命令会跳过大约 20 个文件，报告不能识别这些文件格式，其中大多数是脚本而不是二进制文件。

千万不要在库文件上使用 `--strip-unneeded` ，否则会破坏其静态版本，这样你不得不又从头开始编译全部的工具链软件包。

删除文档还可以节省 20 MB 空间：

```
rm -rf /tools/{info,man}
```

现在 `$LFS` 上就有至少 850 MB 剩余空间，可以在下一章编译安装 Glibc 。如果有足够空间编译安装 Glibc ，那编译安装其它的软件包也就没有问题了。

5.32. 改变所有者

注意

本书剩余部分的命令必须以 `root` 用户登陆后执行而不再使用 `lfs` 用户了。同样，有必要再一次检查 `root` 用户环境下的 `$LFS` 环境变量是否被正确的设置了。

目前， `$LFS/tools` 目录的所有者是仅存在于宿主环境中的 `lfs` 用户。如果保留该目录，那么该目录内文件的所有者的 user ID 就没有对应的账号。这会带来安全上的问题，在以后创建一个用户帐号的时候，如果该用户帐号的 user ID 刚好与目录 `$LFS/tools` 所有者的 user ID 相同，那么该目录下的文件就会面临被恶意操作的危险。

为了避免这个问题，在后面建立 LFS 系统的时候，在创建 `/etc/passwd` 文件时添加与宿主系统的 user ID 和 group ID相同的 `lfs` 用户。另外一个更好的办法是通过下面的指令把 `$LFS/tools` 目录以及其中文件的所有者改为 `root` 用户：

```
chown -R root:root $LFS/tools
```

虽然在 LFS 系统完成的时候，可以把 `$LFS/tools` 目录删除，但是它还可以再用来建立多个相同版本的 LFS 系统，所以很多人会选择保留该目录。如何以最好的方法备份 `$LFS/tools` 目录取决于个人喜好，我们把这个任务留给读者作为练习。

III. 构建 LFS 系统

第六章 安装系统基础软件

6.1. 简介

在这一章，我们进入"建筑工地"，开始精心构建 LFS 系统。也就是指我们通过 chroot 命令进入一个临时的微型 Linux 系统，并作一些最后的准备，然后开始安装软件包。

软件的安装非常直截了当。尽管许多情况下对安装过程的说明可以更加简洁，但为了消除可能出现的错误，我们为每一个包都提供了全面的安装说明。理解 Linux 系统是如何工作的关键在于明白每个包的用途以及为什么用户(或系统)需要它。对每一个安装的软件包，我们都给出了对其内容的简要说明，另外，对该软件包所安装的每一个程序文件和库文件也作了简要的描述。

如果在本章中进行编译器优化，那么请看看编译器优化提示：*http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*，也可以看看 LinuxSir 上的一篇帖子：GCC 编译选项及优化提示。编译器优化可以使程序运行的稍快一些，但也会出现某些编译问题。如果某个包在使用优化的情况下无法通过编译，试试不用优化编译能不能解决问题。即使使用优化编译成功，由于源码与编译工具之间复杂的相互作用，程序仍有可能被错误的编译了。要注意 `-march` 和 `-mtune` 选项或许会导致一些工具链软件包(Binutils, GCC, Glibc)的问题。使用编译器优化得到的小幅度性能提升，与它带来的风险相比微不足道。所以初次编译 LFS 的用户最好不要使用任何优化，你的系统依然会又快又稳定。

本章中软件包的安装顺序应当严格遵守，以确保没有一个程序会把 /tools 作为路径硬连接到代码中。同样不要并行编译包。并行编译可能会节省时间(特别是在双CPU的机器上)，但也可能造成程序包含 /tools 硬连接路径，以致在 /tools 目录被删除之后，程序无法运行。

在每个软件包安装说明页的首部都提供了与该包相关的一些信息，包括：包内容的简要说明、编译大约所需时间、编译过程所需磁盘空间、编译所依赖的软件包。在安装说明之后还有该包所安装的程序和库的列表以及对它们的简要说明。

6.2. 挂载虚拟内核文件系统

虚拟内核文件系统(Virtual Kernel File Systems)，是指那些是由内核产生但并不存在于硬盘上(存 在于内存中)的文件系统，他们被用来与内核进行通信。

首先让我们为虚拟内核文件系统建立挂载目录：

```
mkdir -pv $LFS/{dev,proc,sys}
```

6.2.1. 创建初始设备节点

内核在引导时要求某些设备节点必须存在(特别是 console 和 null)，这些设备节点必须创建在硬盘 上才能使得内核在 udev 尚未启动之前就可以使用它们，此外还有当Linux 以 `init=/bin/bash` 启动。使用下面的命令来创建这些节点：

```
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3
```

6.2.2.挂载并填充 /dev 目录

推荐的向 /dev 目录填充设备的方法是在 /dev 上挂载一个虚拟文件系统(比如 tmpfs)，然后在设备 被检测到或被访问到的时候(通常是在系统引导的过程中)动态创建设备节点。既然现在新的系统尚未被引导， 那么就有必要通过手工挂载和填充 /dev 目录。这可以通过绑定挂载宿主系统的 /dev 目录。绑定挂载是一种 特殊的挂载方式，允许你创建一个目录或者是挂载点的镜像到其他的地方。可以使用下面的命令：

```
mount --bind /dev $LFS/dev
```

6.2.3. 挂载虚拟内核文件系统

现在挂载虚拟内核文件系统：

```
mount -vt devpts devpts $LFS/dev/pts
mount -vt tmpfs shm $LFS/dev/shm
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
```

6.3. 包管理

对于 LFS BOOK，包管理通常被请求加进去。一个包管理器允许跟踪文件的安装，使删除或升级软件包变得简单。在这一部分里，我们不会讨论或者是推荐任何一个包管理器。我们讲述的是一些流行的技术，以及他 们是怎么工作的。对于你来说，一个完美的包管理器可能在这些技术之中，也可能是一些技术的结合。这一部分 简明的描述了当升级软件包的时候会出现的几个问题。

LFS 和 BLFS 中没有涉及包管理器的几个原因有：

- 把精力拿来做包管理器，就偏离了我们的LFS的初衷：一个Linux系统是如何构建的。
- 包管理有很多解决方案，每一个都有优点和缺点。任何一个都不会令其他种类的fans满意的。

有一些hints是关于包管理。可以查看 *Hints subproject* 来选择一个适合你的。

6.3.1. 升级问题

一个包管理器会使升级一个软件包到新的版本变得很简单。通常 LFS 和 BLFS 手册中的说明可以用来升级软件包。 这里有几点，你在升级软件包的时候应该注意的，尤其是在一个运行着的系统上。

- 如果工具链中的一个软件包 (Glibc, GCC 或 Binutils)需要升级到一个新的次版本，重新构建LFS是比较安全的。 尽管你可以按照依赖关系，只是重新构建一部分的软件包。但我们并不推荐这样做。例如，如果 glibc-2.2.x 需要 升 级到 glibc-2.3.x,重新构建是比较安全的。对于小版本的升级，简单的重新安装通常可以正常的工作，但是不能够保证。 例如，从 glibc-2.3.4 升级到 glibc-2.3.5 通常不会有问题。
- 如果包含一个共享库的软件包升级，并且共享库的名字发生了变化。动态链接到这个库的所有的 包都需要重新编译链接到新的库（注意包的版本和库的名字没有关系）。例如，一个软件包 foo-1.2.3 安装了一个名为 `libfoo.so.1` 的共享库。你升级这个包到新版本 foo-1.2.4，这个新版本安装名为 `libfoo.so.2` 的共享库。这种情况下，所有链接到 `libfoo.so.1` 的包都需要重新编译链接到 `libfoo.so.2` 。注意在依赖的软件包没有编译完之前，不要删除原来的库。

6.3.2. 包管理技术

下面是一些常用的包管理的技术。在决定安装包管理器之前，对多种包管理技术进行一下研究，找出某一种的缺点。

6.3.2.1. 全部记在心里！

是的，这是一种包管理的技术。一些人不需要包管理器，因为他们对包都非常熟悉，知道每一个包所安装的文件。 一些人也不需要包管理器，是因为当一个软件包改变时，他们重新构建整个系统。

6.3.2.2. 分别安装到不同目录

这是一种最简单的包管理方式，不需要安装额外的软件包来管理安装过程。每一个软件包安装到不同的目录下。 例如，包 foo-1.1 安装到 `/usr/pkg/foo-1.1` ，创建一个从 `/usr/pkg/foo` 指向 `/usr/pkg/foo-1.1` 的符号链接。当安装一个新版本 foo-1.2 时，它会安装在 `/usr/pkg/foo-1.2` ，前面的符号链接也指向新版本。

一些环境变量，像 `PATH`，`LD_LIBRARY_PATH`，`MANPATH`，`INFOPATH` 和 `CPPFLAGS` 都需要增加 `/usr/pkg/foo` 。当软件包数量大了之后，就难于管理了。

6.3.2.3. 符号连接风格的包管理

这是前面包管理技术的一个变种。每一个包也是按照类似于前面方法进行安装。但是不是创建符号链接， 而是每一个都被链接到 `/usr` 目录。这样就不需要添加环境变量了。尽 管有时在安装的时候符号链接自动创建，还是有很多的是采用这种方法的。一些比较流行的有：Stow，Epkg，Graft，和 Depot。

安装过程需要伪造，这样包就会认为自己被安装到了 `/usr` 目录，尽管实际上 它们安装到 `/usr/pkg` 目录下。以这种风格安装并不是很麻烦的事情。比如，你要 安装一个包 libfoo-1.1 。用下面的指令安装就不合适：

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

安装是没有问题，但是依赖的包不能够像你想像的那样链接到 libfoo。如果你编译一个链接到 libfoo 的包， 你就要注意，要链接到 `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` ，而不是 像你想像的那样链接到 `/usr/lib/libfoo.so.1`。正确的方法是使用 `DESTDIR` 来伪造包的安装。可以这样来使用这种方法：

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

大多数的包都支持这种方法，但还是有一些不支持。对于这些不兼容的包，你可以手工安装，或许把一些有问题的包 安装到 `/opt` 可能更简单。

6.3.2.4. 基于时间戳

这种技术里，在安装包之前，会创建一个时间戳标记文件。安装之后，简单的使用的 `find` 命令的某些选项就能生成一个时间戳标记文件创建之后安装的文件的日志。install-log 就是利用这种技术写的包管理器。

尽管这种方法简单，但是它有两个缺点。如果在安装的过程中，安装的文件的时间戳不是当前的时间，这些文件将不会被包管理器记录。 另外，这种方案只能用在一次只有一个包安装时。如果有两个包在两个终端下同时安装，那么这时的日志是不可靠的。

6.3.2.5. 基于LD_PRELOAD

这种方法下，在安装之前会有一个库被提前加载。在安装的过程中，这个库就会跟踪正在安装的包，通过给自己附加上 各种可执行性的动作，像 `cp` , `install` , `mv` 。来跟踪那些修改文件系统系统调用。为了让这种方法正 常的工作，所有的可执行文件需要都是动态链接的且没有设置 suid 和 sgid 位。预先加载库可能会产生一些讨厌的 副作用。因此，建议做一些测试，来确保包管理器不会破坏任何东西，并且能够记录所有适当的文件。

6.3.2.6. 创建包的归档

在这种方案中，包被分别安装到不同的目录树下，就像符号连接风格的包管理描述的。安装之后， 系统就会使用安装的文件创建一个包的归档。这个档案可以在本机上安装包，甚至可以在其他机器上 安装包。

这种方法被大多数商业发行版的包管理器采用。例如，RPM（自然是*Linux Standard Base Specification*所必需），pkg-utils，Debian 的 apt 和 Gentoo 的 Portage 系统。 有一个描述在 LFS 系统中如何应用这样的包管理器的hint，参见 *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

6.3.2.7. 基于用户的管理

这种方案是 LFS 特有的，是由 Matthias Benkmann 提出的，可以参考 *Hints Project*。在这种方案里，每一个包都是以不同的用户安装到标准的目录里。通过检验 user ID 可以很容易的标识一个软件包。这种方法的特点和缺点非常复杂，这里就不描述了。详细内容参见 *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*。

6.4. 进入 Chroot 环境

现在将要进入 chroot 环境开始编译安装最终的 LFS 系统了，注意：在这里我们只使用临时构建的工具。以 root 身份运行以下命令进入构建环境：

```
chroot "$LFS" /tools/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin:/tools/bin \
    /tools/bin/bash --login +h
```

env 命令的参数 -i 的作用是清除所有 chroot 环境变量。后面是重新设定 HOME, TERM, PS1, PATH 等变量的值。TERM=$TERM 设定虚拟根环境中的 TERM 的值与 chroot 外面的一样。这个值是让像 vim 和 less 之类的程序可以正确操作。如果还需要重新设置其它的值，如 CFLAGS 或 CXXFLAGS ，这里是个不错的位置。

从这里开始，不再需要 LFS 环境变量了，因为所有的工作都被限制在 LFS 文件系统里面。这是由于已经告诉了 Bash shell $LFS 是现在的根目录(/)。

注意，这里 /tools/bin 位于 PATH 的最后面。也就是说当软件包的最终版本安装之后就不再使用临时工具了。为了使 shell 无法"记住"可执行二进制代码的位置，需要通过使用 +h 参数关闭 bash 的散列功能。

注意此时 bash 提示符会显示：I have no name! 这是正常的，因为 /etc/passwd 还没有创建。

注意

这一章剩下的命令以及后面几章的命令都是在chroot环境下进行的。如果你离开了这个环境（比如重启）， 要确保内核虚拟文件系统挂载，像在 节 6.2.2, "挂载并填充 /dev 目录" 和 节 6.2.3, "挂载虚拟内核文件系统" 描述的。在继续安装之前，再次进入chroot环境。

6.5. 创建系统目录结构

现在我们在 LFS 分区中创建目录树结构，用下列命令能创建一个标准的目录树：

```
mkdir -pv /{bin,boot,etc/opt,home,lib,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
  ln -sv share/{man,doc,info} $dir
done
mkdir -v /var/{lock,log,mail,run,spool}
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

缺省的目录的权限模式为 755，但也并非所有的目录都如此。以上的命令有两处有所不一样：一个是 root 用户的目录，另外两个是临时文件目录。

第一个权限模式的不同之处是确保禁止任何人进入到 /root 目录中——同样的，这个模式也适用于让其它的普通用户可以工作在自己的目录中。第二个权限模式的不同之处是确保所有用户都可以写 /tmp 和 /var/tmp 目录，但不能从中删除其它用户的文件，这是由"sticky位"，也就是"1777"中的"1"来设定的。

6.5.1. FHS 兼容性说明

我们的目录树是按照 FHS(Filesystem Hierarchy Standard) 标准 (http://www.pathname.com/fhs/)。除了上面创建的目录外，该标准还规定了必须有 /usr/local/games 和 /usr/share/games 两个目录，但是作为一个基本系统，我们并不需要这些。如果你要完全的遵守 FHS 标准的话，就自己建立这两个目录。至于 /usr/local/share 目录下的子目录，FHS 标准规定得并不严格，所以我们就创建了(在我们看来)需要的子目录。

6.6. 创建必需的文件与符号连接

一些程序使用固化的路径(hard-wired paths)指向一些目前还不存在的程序上。为了兼容这些程序，可以创建一些符号链接，然后在软件安装之后用实际文件进行替代。

```
ln -sv /tools/bin/{bash,cat,grep,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv bash /bin/sh
```

一个常规的Linux系统在 `/etc/mtab` 中有一个已挂载文件系统的列表。 正常情况下，这个文件在我们挂载一个新的文件系统的时候会被创建。因为我们在chroot环境下不会再挂载任何文件系统 ，所以我们需要为那些用到 `/etc/mtab` 的程序创建一个空文件：

```
touch /etc/mtab
```

为了让 root 用户可以登录而且用户名"root"可以被识别，在这里需要创建相应的 /etc/passwd 和 /etc/group 文件。

使用下面的命令创建 /etc/passwd 文件：

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
EOF
```

root 的真正密码将在后面设置("x"在这里只是一个占位符)。

下面的命令创建 /etc/group 文件：

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
EOF
```

这里创建的用户组并不是某个标准所要求的部分，只是因为在随后 Udev 配置将要用到而以。Linux 标准基础(LSB, Linux Standard Base, *http://www.linuxbase.org*) 只是推荐"root"组的 GID 为 0，另一个组"bin"的 GID 为 1 。其它所有的组名和 GID 均由系统管理员自由设定，因为比较好的软件包一般都不依赖于 GID ，而只是使用组名。

因为完整的 Glibc 在Chapter 5中已经安装，而且 /etc/passwd 和 /etc/group 文件也已创建，用户名和组名现在可以开始使用了。现在启动新的shell，驱除"I have no name!"提示符：

```
exec /tools/bin/bash --login +h
```

注意这里使用了 +h 参数。这是告诉 bash 不能使用其内部哈希表查找路径。如果没有使用这个参数，则 bash 将会记住已经执行的二进制代码的路径。为了让新编译安装的二进制代码可以马上投入使用，在这一章中，我们使用 +h 关闭了此功能。

程序 login, agetty, init (还有其它一些程序) 使用一些日志文件来记录信息，比如谁在什么时候登录了系统等等。然而如果这些日志文件不存在，这些程序则无法写入。下面初始化这些日志文件，并设置适当的权限：

```
touch /var/run/utmp /var/log/{btmp,lastlog,wtmp}
chgrp -v utmp /var/run/utmp /var/log/lastlog
chmod -v 664 /var/run/utmp /var/log/lastlog
```

/var/run/utmp 记录着现在登录的用户。/var/log/wtmp 记录所有的登录和退出。/var/log/lastlog 记录每个用户最后的登录信息。/var/log/btmp 记录错误的登录尝试。

6.7. Linux-Libc-Headers-2.6.12.0

Linux-Libc-Headers 包含"纯净的"内核头文件。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 27 MB

6.7.1. 安装 Linux-Libc-Headers

多年来，通常的做法是直接从内核源代码中复制出"原始"内核头文件，放在 /usr/include 中使用。但是最近几年，内核开发人员强烈要求停止这种做法。因此诞生了 Linux-Libc-Headers 项目，其目的就是维护一个 API 版本稳定的内核头文件。

添加一个用户空间头文件和新内核对于inotify特性的系统调用支持:

```
patch -Np1 -i ../linux-libc-headers-2.6.12.0-inotify-3.patch
```

安装内核头文件：

```
install -dv /usr/include/asm
cp -Rv include/asm-i386/* /usr/include/asm
cp -Rv include/linux /usr/include
```

确保这些头文件的所有者是 root ：

```
chown -Rv root:root /usr/include/{asm,linux}
```

确保用户可以读取这些头文件：

```
find /usr/include/{asm,linux} -type d -exec chmod -v 755 {} \;
find /usr/include/{asm,linux} -type f -exec chmod -v 644 {} \;
```

6.7.2. Linux-Libc-Headers 的内容

安装的头文件： /usr/include/{asm,linux}/*.h

简要描述

`/usr/include/{asm,linux}/*.h`	内核头文件 API

6.8. Man-pages-2.34

Man-pages 包含 1200 页的用户手册。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 18.4 MB

6.8.1. 安装 Man-pages

使用下述命令安装 Man-pages ：

```
make install
```

6.8.2. Man-pages 的内容

安装的文件： 一些用户手册文件

简要描述

`man pages`	描述了 C 和 C++ 的函数、重要的设备文件、以及一些重要的配置文件。

6.9. Glibc-2.3.6

Glibc 包含了主要的C库。这个库提供了基本例程，用于分配内存、搜索目录、打开关闭文件、读写文件、字串处理、模式匹配、数学计算等等。

预计编译时间： 13.5 SBU (含测试套件)所需磁盘空间： 510 MB (含测试套件)

6.9.1. 安装 Glibc

注意

一些 LFS 基本系统之外的软件包建议安装 GNU libiconv 以使得数据够能在不同编码之间进行转换。GNU libiconv 项目的主页(*http://www.gnu.org/software/libiconv/*) 说："这个库为那些没有 iconv() 的系统或者虽然有 iconv() 却不能与 Unicode 相互转换的系统提供了一个能够与 Unicode 相互转换的实现 "。Glibc 库中有一个 iconv() ，并且能够与 Unicode 相互转换，因此，LFS 系统不需要 GNU libiconv 。

Glibc 的编译系统是高度自给自足的，即使当前编译器 specs 文件和连接器还指向 /tools 目录，也能正确安装。我们在安装 Glibc 前不能调整 specs 文件和连接器，否则 Glibc 的 autoconf 测试会失败，从而妨碍我们创建一个干净系统的目标。

glibc-libidn这个包加上了对国际化域名(IDN)的支持到Glibc中。许多程序支持IDN需要全的 `libidn` 库，而不是这个附加库。(参见 *http://www.linuxfromscratch.org/blfs/view/svn/general/libidn.html*).

解压缩包到Glibc的源码目录：

```
tar -xf ../glibc-libidn-2.3.6.tar.bz2
```

应用下面这个patch来修正软件包在 `sys/kd.h` 之后包含 `linux/types.h` 导致编译错误：

```
patch -Np1 -i ../glibc-2.3.6-linux_types-1.patch
```

添加一个头文件来定义为新内核对于inotify特性的系统调用函数：

```
patch -Np1 -i ../glibc-2.3.6-inotify-1.patch
```

在vi_VN.TCVN locale中, `bash` 一启动就进入一个无限循环之中。不知道这是一个 `bash` 的 bug，还是一个Glibc的问题。为了避免这个问题，抑制这个locale的安装：

```
sed -i '/vi_VN.TCVN/d' localedata/SUPPORTED
```

当运行 **make install**，一个叫 `test-installation.pl` 的脚本会在我们新安装的Glibc上做一个小的完整性测试。然而，由于我们的toolchain仍然指向 `/tools` 目录，完整性测试会导致使用错误的Glibc。我们可以强制脚本测试我们刚安装的脚本：

```
sed -i \
's|libs -o|libs -L/usr/lib -Wl,-dynamic-linker=/lib/ld-linux.so.2 -o|' \
        scripts/test-installation.pl
```

Glibc 文档推荐在源码目录之外的一个专门的编译目录下进行编译：

```
mkdir -v ../glibc-build
cd ../glibc-build
```

为编译 Glibc 做准备：

```
../glibc-2.3.6/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

新配置选项的含义：

`--libexecdir=/usr/lib/glibc`

把 `pt_chown` 程序的位置从默认的 `/usr/libexec` 改为 `/usr/lib/glibc` 。

编译软件包：

```
make
```

重要

本节的 Glibc 测试很重要。在任何情况下都不要省略这一步。

对结果进行测试：

```
make -k check 2>&1 | tee glibc-check-log
grep Error glibc-check-log
```

在*posix/annexc* 中，你可能会看到一个预料的错误（可以忽略）。另外，Glibc测试单元，多少依赖于宿主系统。下面是一些常见的错误：

- *nptl/tst-clock2* 和*tst-attr3* 测试有时会出错。 原因现在还不是很明白，可能是系统负载过重导致的。

- *math* 测试在一些使用较老的 Intel 或 AMD 的系统上会失败，某些优化设置也会导致该测试失败。
- *atime* 会在使用 `noatime` 选项挂载 LFS 分区时失败（参见节 2.4, "挂载新分区"），在编译LFS过程中不要使用 `noatime` 选项。
- 在一些很老很慢的机器上，一些测试会由于超时而失败。

在安装 Glibc 的过程中，它会警告缺少 `/etc/ld.so.conf` 文件。其实这没什么关系，不过下面的命令能修正它：

```
touch /etc/ld.so.conf
```

安装软件包：

```
make install
```

安装inotify头文件到系统头文件的地方：

```
cp -v ../glibc-2.3.6/sysdeps/unix/sysv/linux/inotify.h \
    /usr/include/sys
```

locales能使系统以一种上面命令没有安装的语言处理。要注意locales是必须的，如果他们中的一些丢失了，后面的测试单元会跳过重要测试。

单个的locale可以通过使用 `localedef` 程序来安装。例如，下面的第一个 `localedef` 命令将 `/usr/share/i18n/locales/de_DE` 跟 `/usr/share/i18n/charmaps/ISO-8859-1.gz` 结合，并添加到 `/usr/lib/locale/locale-archive` 文件中。下面的说明将会安装一个所需locale的最小集合：

```
mkdir -pv /usr/lib/locale
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

另外，你可以安装你的国家、语言和字符集所对应的locale。

当然，可以一次安装所有列在 `glibc-2.3.6/localedata/SUPPORTED` 中的locales，利用下面的命令：

```
make localedata/install-locales
```

接着使用 `localedef` 命令来创建和安装locales没有列在 `glibc-2.3.6/localedata/SUPPORTED` 中的（这种情况不太可能），如果你需要他们的话。

6.9.2. 配置 Glibc

我们需要建立 /etc/nsswitch.conf 文件。因为在这个文件丢失或不正确的情况下，Glibc 会使用默认配置，而 Glibc 的默认配置无法很好地在网络环境下工作。并且我们也需要设置自己的时区。

使用如下命令建立一个新的 /etc/nsswitch.conf 文件：

```
cat > /etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# End /etc/nsswitch.conf
EOF
```

要想确定本地时区，可以使用下面的脚本：

```
tzselect
```

按照顺序回答脚本运行过程中提出的几个问题后，脚本就会给出所需时区文件的位置(比如 *America/Edmonton*)。还有其他的一些时区列在 `/usr/share/zoneinfo` 中，比如 *Canada/Eastern* 或 *EST5EDT*，这些虽然没有被脚本识别，但是都可以使用。

再用下列命令来创建 /etc/localtime 文件：

```
cp -v --remove-destination /usr/share/zoneinfo/<xxx> \
    /etc/localtime
```

将 `<xxx>` 替换成选择的时区的名称(比如，Canada/Eastern)。

cp 命令选项的含义：

`--remove-destination`

强制删除已存在的符号链接。我们采用拷贝文件而不是创建符号链接的原因是：有可能 /usr 在单独的分区上，如果启动进入单用户模式，就会出问题。

6.9.3. 配置动态链接库加载程序

默认情况下，动态链接库加载程序(/lib/ld-linux.so.2)搜索 /lib 和 /usr/lib 目录来寻找程序需要使用的动态连接库。但是，如果某些库在这两个目录之外，你就需要把它们的路径加到 /etc/ld.so.conf 文件里，以便动态链接库加载程序能够找到它们。 /usr/local/lib 和 /opt/lib 是两个经常包含动态连接库但又不在默认目录中的目录，我们要把它们添加到动态链接库加载程序的搜索路径中。

使用如下命令创建新的 /etc/ld.so.conf 文件：

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf

/usr/local/lib
/opt/lib

# End /etc/ld.so.conf
EOF
```

6.9.4. Glibc 的内容

安装的程序： catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, nscd_nischeck, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump, zic安装的库： ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.a, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so, libutil.{a,so}

简要描述

`catchsegv`	当程序发生段故障的时候，用来建立一个堆栈跟踪
`gencat`	建立消息列表
`getconf`	针对文件系统的指定变量显示其系统设置值
`getent`	从系统管理数据库获取一个条目
`iconv`	字符集转换
`iconvconfig`	建立快速加载的 iconv 模块所使用的配置文件

`ldconfig`	配置动态链接库的实时绑定
`ldd`	列出每个程序或者命令需要的共享库
`lddlibc4`	帮助 ldd 操作目标文件
`locale`	打印当前locale的详细信息
`localedef`	编译 locale 标准
`mtrace`	读取并解释一个内存跟踪文件然后以人类可读的格式显示一个摘要
`nscd`	为最常用的名称服务请求提供缓存的守护进程
`nscd_nischeck`	检查在进行 NIS+ 查找时是否需要安全模式
`pcprofiledump`	转储 PC profiling 产生的信息
`pt_chown`	一个辅助程序，帮助 grantpt 设置子虚拟终端的属主、用户组、读写权限
`rpcgen`	产生实现远程过程调用(RPC)协议的 C 代码
`rpcinfo`	对 RPC 服务器产生一个 RPC 呼叫
`sln`	ln 程序使用静态连接编译的版本，在 ln 不起作用的时候，sln 仍然可以建立符号链接
`sprof`	读取并显示共享目标的特征描述数据
`tzselect`	对用户提出关于当前位置的问题并输出时区信息到标准输出
`xtrace`	通过打印当前执行的函数跟踪程序执行情况
`zdump`	显示时区
`zic`	时区编译器
`ld.so`	帮助动态链接库执行的辅助程序
`libBrokenLocale`	帮助应用程序(如Mozilla)处理破损的 locale
`libSegFault`	段故障信号处理器
`libanl`	异步名称查询库
`libbsd-compat`	为了在 linux 下执行一些 BSD 程序，libbsd-compat 提供了必要的可移植性
`libc`	主 C 库，集成了最常用函数
`libcidn`	被Glibc使用，在 `getaddrinfo()` 函数中来处理国际域名
`libcrypt`	用于的加密库
`libdl`	动态连接接口库
`libg`	g++ 运行时库
`libieee`	IEEE 浮点运算库
`libm`	数学函数库

`libmcheck`	包括了启动(boot)时需要的代码
`libmemusage`	帮助 memusage 搜集程序运行时的内存占用信息
`libnsl`	提供网络服务的库
`libnss`	名称服务切换库，包含了解释主机名、用户名、组名、别名、服务、协议等等的函数
`libpcprofile`	包含用于跟踪某些特定源代码的 CPU 使用时间的 profiling 函数
`libpthread`	POSIX 线程库
`libresolv`	创建、发送、解释到互联网域名服务器的数据包
`librpcsvc`	提供 RPC 的其他杂项服务
`librt`	提供了大部分的 POSIX.1b 运行时扩展接口
`libthread_db`	对多线程程序的调试很有用
`libutil`	包含了在很多不同的 Unix 程序中使用的"标准"函数

6.10. 再次调整工具链

现在，最终的 C 库已经安装好了，我们需要再次调整工具链，让本章随后编译的那些工具都连接到这个库上。基本上，就是把 Chapter 5 中"调整工具链"那里做的调整给取消掉。在 Chapter 5 中，工具链使用的库是从宿主系统的 `/{,usr/}lib` 转向新安装的 `/tools/lib` 目录。同样的，现在工具链使用的库将从临时的 `/tools/lib` 转向 LFS 系统最终的 `/{,usr/}lib` 目录。

首先，备份 `/tools` 下的链接，用我们在第 5 章中编译的链接器来替换，再创建一个链接到在 `/tools/$(gcc -dumpmachine)/bin` 中的复本。

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

接下来，修正 GCC 的 specs 文件，使它指向新的动态链接器，这样 GCC 才能知道在哪能发现开始文件。应用一个 `perl` 命令：

```
gcc -dumpspecs | \
perl -p -e 's@/tools/lib/ld-linux.so.2@/lib/ld-linux.so.2@g;' \
    -e 's@\*startfile_prefix_spec:\n@$_/usr/lib/ @g;' > \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

修改之后，用你的眼睛亲自检查一下 specs 文件，确保已经改正确了。

重要

如果你的系统平台上，动态连接器的名字不是 `ld-linux.so.2` ，你必须把上面命令里的"ld-linux.so.2"换成你的系统平台上动态连接器的名字。若有必要，请参见 Section 5.2 "工具链技术说明"。

现在有必要停下来，检查一下新工具链的基本功能(编译和连接)是否正常，我们进行一个简单的合理性检查：

```
echo 'main(){}' > dummy.c
cc dummy.c -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

如果一切正常，应该不会出错，而且最后一个命令的结果应该是(某些特殊平台上动态连接器的名称可能与此处不同)：

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

注意，`/lib` 应该是动态连接器的前缀。

现在确保我们设置使用正确的开始文件：

```
grep -o '/usr/lib.*/crt[1in].* .*' dummy.log
```

如果一切正常，应该不会出错，而且最后一个命令的结果应该是：

```
/usr/lib/crt1.o succeeded
/usr/lib/crti.o succeeded
/usr/lib/crtn.o succeeded
```

接下来要做的是验证新的链接器是否在正确的搜索路径内：

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

如果一切正常，应该不会出错，而且最后一个命令的结果应该是：

```
SEARCH_DIR("/tools/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

下面，确保我们是否正在使用正确的 libc：

```
grep "/lib/libc.so.6 " dummy.log
```

如果一切正常，应该不会出错，而且最后一个命令的结果应该是：

```
attempt to open /lib/libc.so.6 succeeded
```

最后，确保 GCC 正在使用正确的动态链接器：

```
grep found dummy.log
```

如果一切正常，应该不会出错，而且最后一个命令的结果应该是(某些特殊平台上动态连接器的名称可能与此处不同)：

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

如果输出与上面不同或者没有输出，那么就有大问题了。你需要检查一下前面的操作，看看问题出在哪里，并改正过来。在改正之前，不要继续后面的部份，因为没什么意义。大多数情况下，出错都是因为上面的 specs 文件没改对。当然，如果你的平台上动态连接器的名字不是 ld-linux.so.2 ，上面的结果也会不同。在继续之前要解决所有的问题。

在确定一切正常后，删除测试文件：

```
rm -v dummy.c a.out dummy.log
```

6.11. Binutils-2.16.1

Binutils 是一组开发工具，包括连接器、汇编器和其他用于目标文件和档案的工具。

预计编译时间： 1.5 SBU (含测试套件)所需磁盘空间： 172 MB (含测试套件)

6.11.1. 安装 Binutils

现在我们测试一下在 chroot 环境中，你的伪终端(PTY)是否正常工作。运行下面的命令：

```
expect -c "spawn ls"
```

如果看到这样的输出：

```
The system has no more ptys.
Ask your system administrator to create more.
```

说明你的chroot环境还没有设置好 PTY，这时运行 Binutils 和 GCC 的测试套件就没有意义了，你必须先解决 PTY 设置。

Binutils 的文档推荐用一个新建的目录来编译它，而不是在源码目录中：

```
mkdir -v ../binutils-build
cd ../binutils-build
```

为编译 Binutils 做准备：

```
../binutils-2.16.1/configure --prefix=/usr \
    --enable-shared
```

编译软件包：

```
make tooldir=/usr
```

make 参数的含义：

`tooldir=/usr`

通常情况下，tooldir(可执行文件的安装目录) 是 `$(exec_prefix)/$(target_alias)` 。例如在 i686 机器上，将是 tt class="filename">/usr/i686-pc-linux-gnu 。因为我们只为自己的系统进行编译，就并不需要在 `/usr` 目录后面再存在特殊的后缀。 `$(exec_prefix)/$(target_alias)` 只是在交叉编译时(比如在 Intel 机器上编译将要在 PowerPC 上执行的程序)才用到。

重要

本节的 Binutils 测试套件很重要。在任何情况下都不要省略这一步。

对结果进行测试：

```
make check
```

安装软件包：

```
make tooldir=/usr install
```

安装某些软件包需要的 `libiberty` 头文件：

```
cp -v ../binutils-2.16.1/include/libiberty.h /usr/include
```

6.11.2. Binutils 的内容

安装的程序： addr2line, ar, as, c++filt, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings, strip安装的库： libiberty.a, libbfd.{a,so}, libopcodes.{a,so}

简要描述

`addr2line`	把程序地址转换为文件名和行号。在命令行中给它一个地址和一个可执行文件名，它就会使用这个可执行文件的调试信息指出在给出的地址上是哪个文件以及行号。
`ar`	建立、修改、提取归档文件。归档文件是包含多个文件内容的一个大文件，其结构保证了可以恢复原始文件内容。
`as`	一个汇编器，用来汇编 `gcc` 的输出，产生的目标文件然后由接器 ld 连接。
`c++filt`	连接器使用它来过滤 C++ 和 Java 符号，防止重载函数冲突。
`gprof`	显示程序调用段的各种数据。
`ld`	连接器，它把一些目标和归档文件结合为一个文件，重定位数据，并链接符号引用。通常，建立一个新编译程序的最后一步就是调用 ld 。
`nm`	列出出现在目标文件中的符号
`objcopy`	把一种目标文件中的内容复制到另一种类型的目标文件中
`objdump`	显示所给目标文件的信息。使用选项来控制其显示的信息。它所显示的信息通常只有编写编译工具的人才感兴趣。
`ranlib`	产生归档文件索引，并将其保存到这个归档文件中。在索引中列出了归档文件各成员所定义的可重分配目标文件。
`readelf`	显示 ELF 格式可执行文件的信息
`size`	列出目标文件每一段的大小以及总体的大小。默认情况下，对于每个目标文件或者一个归档文件中的每个模块只产生一行输出。
`strings`	打印某个文件的可打印字符串，这些字符串最少 4 个字符长，也可以使用选项"-n"设置字符串的最小长度。默认情况下，它只打印目标文件初始化和可加载段中的可打印字符；对于其它类型的文件它打印整个文件的可打印字符，这个程序对于了解非文本文件的内容很有帮助。
`strip`	删除目标文件中的全部或者特定符号
`libiberty`	包含许多GNU程序都会用到的函数，这些程序有： `getopt` ， `obstack` ， `strerror` ， `strtol` ， 和 `strtoul`
`libbfd`	二进制文件描述库
`libopcodes`	用来处理 opcodes（"可读文本格式的"）处理器操作指令)的库，在生成一些应用程序的时候也会用到它，比如 `objdump` 。

6.12. GCC-4.0.3

GCC 软件包包含 GNU 编译器，其中有 C 和 C++ 编译器。

预计编译时间： 22 SBU (含测试套件)所需磁盘空间： 566 MB (含测试套件)

6.12.1. 安装 GCC

使用一个 `sed` 命令来禁止 GCC 安装它自己的 `libiberty.a` 。我们将使用 Binutils 附带的 `libiberty.a` 来代替：

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

在节 5.4, "GCC-4.0.3 - 第一遍"中应用的 bootstrap 编译中，编译器会有 `-fomit-frame-pointer` 的标志。非bootstrap编译默认是忽略这个标志的，可以应用下面的 `sed` 命令来确保编译的可靠性。

```
sed -i 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in
```

`fixincludes` 脚本偶尔会因为修改系统的头文件而出错。因为GCC-4.0.3 和 Glibc-2.3.6 是不需要修改的，运行下面的命令可以避免 `fixincludes` 脚本运行：

```
sed -i 's@\./fixinc\.sh@-c true@' gcc/Makefile.in
```

GCC 中提供了一个 `gccbug` 脚本，会在编译时侦测 mktemp 是否存在，并且在测试中加强代码。这将会导致脚本使用一些不算很随机的名字来命名临时文件。因为我们后面会安装 mktemp ，这里我们就模仿它的存在：

```
sed -i 's/@have_mktemp_command@/yes/' gcc/gccbug.in
```

GCC 的安装指南推荐用一个新建的目录来编译它，而不是在源码目录中：

```
mkdir -v ../gcc-build
cd ../gcc-build
```

为编译 GCC 做准备：

```
../gcc-4.0.3/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-clocale=gnu --enable-languages=c,c++
```

编译软件包：

```
make
```

重要

本节的GCC测试套件很重要。在任何情况下都不要省略这一步。

运行测试套件，但遇到错误不停止(你还记得那些老是出错的测试吧)：

```
make -k check
```

要查看测试单元的测试结果，可以运行：

```
../gcc-4.0.3/contrib/test_summary
```

如果只想看概要，可以把输出通过管道传递给 **`grep -A7 Summ`** 。

结果可以跟 *http://www.linuxfromscratch.org/lfs/build-logs/6.2/* 的进行比较。

一些预想不到的错误总是无法避免的。虽然GCC的开发者经常留意这些问题，但是有些还是没有得到解决。通常， `libmudflap` 测试尤其被认为是有问题的（这是一个bug，参见 *http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*）。除非测试结果给上面URL的有很多不同，一般是可以安全继续的。

安装软件包：

```
make install
```

有的软件包希望 C PreProcessor(预处理器)安装在 `/lib` 目录下，为了满足它们的要求，我们创建如下符号链接：

```
ln -sv ../usr/bin/cpp /lib
```

许多软件包使用 `cc` 作为 C 编译器的名字，为了满足它们的要求，创建如下符号链接：

```
ln -sv gcc /usr/bin/cc
```

现在，我们的最终工具链已经形成了，我们需要做的就是确保编译、链接按照我们希望的完成。我们可以通过本章前面的简册方法来实现：

```
echo 'main(){}' > dummy.c
cc dummy.c -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

如果所有的都工作正常，就不会有错误，并且命令的输出应该是（允许不同平台的动态链接器的名称不同）：

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

现在确保我们使用正确的 startfiles：

```
grep -o '/usr/lib.*/crt[1in].* .*' dummy.log
```

如果所有的都工作正常，就不会有错误，并且命令的输出应该是：

```
/usr/lib/gcc/i686-pc-linux-gnu/4.0.3/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.0.3/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.0.3/../../../crtn.o succeeded
```

接下来，确认新的链接器被应用到了正确的搜索路径中：

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

如果所有的都工作正常，就不会有错误，并且命令的输出应该是：

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

现在，确保我们正在使用正确的 libc ：

```
grep "/lib/libc.so.6 " dummy.log
```

如果所有的都工作正常，就不会有错误，并且命令的输出应该是

```
attempt to open /lib/libc.so.6 succeeded
```

最后，确保 GCC 正在使用正确的动态链接器：

```
grep found dummy.log
```

如果所有的都工作正常，就不会有错误，并且命令的输出应该是（允许不同平台的动态链接器的名称不同）：

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

如果输出不是像上面那样或者根本没有输出，那么就有大问题了。返回并检查前面的操作，找出问题，并改正过来。最有可能的原因是上面修正 specs 文件时出错。任何一个问题都需要在继续之前解决掉。

一旦工具都工作正常，清理测试文件：

```
rm -v dummy.c a.out dummy.log
```

6.12.2. GCC 的内容

安装的程序： c++, cc(→gcc), cpp, g++, gcc, gccbug, gcov安装的库： libgcc.a, libgcc_eh.a, libgcc_s.so, libstdc++.{a,so}, libsupc++.a

简要描述

`cc`	C 编译器
`cpp`	C 预处理器。编译器用它来将 #include 和 #define 这类声明在源文件中展开。
`c++`	C++ 编译器
`g++`	C++ 编译器
`gcc`	C 编译器
`gccbug`	一个shell脚本，帮助创建有价值的 bug 报告。
`gcov`	覆盖测试工具，用来分析在程序的哪里做优化的效果最好。
`libgcc`	`gcc` 的运行时库
`libstdc++`	准 C++ 库，包含许多常用的函数。
`libsupc++`	为 C++ 语言提供支持的库函数。

6.13. Berkeley DB-4.4.20

Berkeley DB 包含一些程序和工具，供其他的一些程序来在做数据库相关函数时调用。

预计编译时间： 1.2 SBU所需磁盘空间： 77 MB

Other Installation Possibilities

如果你需要建立一个 RPC 服务器或者是附加语言绑定编译，在BLFS手册中有一些编译这个软件包的说明。附加语言的绑定编译还需要一些额外的软件包。参见 *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db*的安装说明

另外，GDBM 可以 被用来代替 Berkeley DB 来满足数据库需求。但是，因为在LFS构建过程中，Berkeley DB 被认为是一个核心部分，无法列出在BLFS手册中把它作为依赖的软件（太多了）。同样，很多时候我们测试的是安装了Berkeley DB的LFS系统，而不是 GDBM。如果你清楚的了解了使用 GDBM 的风险和好处，仍然想要采用它，可以参考BLFS手册中的说明 *http://www.linuxfromscratch.org/blfs/view/svn/general/gdbm.html*

6.13.1. 安装 Berkeley DB

修补软件包来防止一些潜在的陷井时间：

```
patch -Np1 -i ../db-4.4.20-fixes-1.patch
```

为编译 Berkeley DB 做准备：

```
cd build_unix &&
../dist/configure --prefix=/usr --enable-compat185 --enable-cxx
```

配置选项的含义：

--enable-compat185

这个选项指定编译 Berkeley DB 1.85 向上兼容性API。

--enable-cxx

这个选项指定编译 C++ API 库。

编译软件包：

```
make
```

现在测试软件包是没有意义的，因为这将会导致 TCL 捆绑编译。TCL不能被准确的编译，因为 TCL 还是链接到 `/tools` 下的 Glibc，而不是 `/usr` 目录下的Glibc。

安装软件包：

```
make docdir=/usr/share/doc/db-4.4.20 install
```

make 参数的含义：

`docdir=...`

这条安装命令将db的文档安装到正确的位置。/p>

修改安装文件的属主：

```
chown -v root:root /usr/bin/db_* \
    /usr/lib/libdb* /usr/include/db* &&
chown -Rv root:root /usr/share/doc/db-4.4.20
```

6.13.2. Berkeley DB 的内容

安装的程序： db_archive, db_checkpoint, db_deadlock, db_dump, db_hotbackup, db_load, db_printlog, db_recover, db_stat, db_upgrade, db_verify安装的库： libdb.{so,ar}and libdb_cxx.r{o,ar}

简要描述

`db_archive`	打印出不再使用的日志文件路径名
`db_checkpoint`	监视和检查数据库日志的守护进程
`db_deadlock`	当死锁发生时，退出锁定要求
`db_dump`	把数据库文件转换成 `db_load` 能认出的文本文件
`db_hotbackup`	创建 "hot backup" 或者是 "hot failover" 的 Berkeley DB 数据库镜像。
`db_load`	从db_dump产生的文本文件中创建出数据库文件
`db_printlog`	把数据库日志文件转换成人能读懂的文本
`db_recover`	在发生错误后，把数据库恢复到一致的状态
`db_stat`	显示数据库环境统计
`db_upgrade`	把数据库文件转换成新版本的Berkley DB格式
`db_verify`	对数据库文件进行一致性检查
`libdb.{so,a}`	包含db处理相关函数的C库
`libdb_cxx.{so,a}`	包含db处理相关函数的C++库

6.14. Coreutils-5.96

Coreutils 软件包包括一套显示、设置基本系统属性的工具。

预计编译时间： 1.1 SBU所需磁盘空间： 58.3 MB

6.14.1. 安装 Coreutils

通常 `uname` 程序总是有点毛病的，比如 _ `-p` _unknown 的结果。下面的补丁对 Intel 平台的机器能修正这个问题：

```
patch -Np1 -i ../coreutils-5.96-uname-1.patch
```

阻止 Coreutils 安装后面将由别的包安装的程序：

```
patch -Np1 -i ../coreutils-5.96-suppress_uptime_kill_su-1.patch
```

POSIX 要求 Coreutils 的程序即使在多字节环境下也能够识别出字符的边界。下面的这个patch能够解决这个问题以及其他的一些国际化相关的问题：

```
patch -Np1 -i ../coreutils-5.96-i18n-1.patch
```

为了测试应用的patch能够运行，修改文件的权限：

```
chmod +x tests/sort/sort-mb-tests
```

注意

过去，在这个patch里面发现了很多bug。当你向 Coreutils 的维护者发送错误报告的时候，首先确认不应用这个patch错误会不会出现。

现在已经发现在使用 `who -Hu` 时，转换的信息有时会导致缓冲区溢出。增大缓冲区大小：

```
sed -i 's/_LEN 6/_LEN 20/' src/who.c
```

为编译 Coreutils 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

Coreutils 软件包的测试套件对系统进行了某些假设，比如要求有非 root 用户和组，但是我们目前的系统中尚不存在。如果你不想运行测试套件，就直接跳过下面将要进行的测试，直接从"安装软件包"那里继续。

下面的命令为我们做测试前的准备，创建两个 dummy(伪) 组和一个 dummy(伪) 用户：

```
echo "dummy1:x:1000:" >> /etc/group
echo "dummy2:x:1001:dummy" >> /etc/group
echo "dummy:x:1000:1000::/root:/bin/bash" >> /etc/passwd
```

现在已经准备好可以运行测试套件了，首先运行那些需要以 `root` 运行的测试：

```
make NON_ROOT_USERNAME=dummy check-root
```

然后以 `dummy` 用户运行剩余的测试：

```
src/su dummy -c "make RUN_EXPENSIVE_TESTS=yes check"
```

测试结束后，删除 dummy 组和用户：

```
sed -i '/dummy/d' /etc/passwd /etc/group
```

安装软件包：

```
make install
```

把一些程序移动到合适的位置以符合 FHS 标准：

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,hostname,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

一些 LFS-Bootscripts 包中的脚本依赖于 `head` ， `sleep` ，和 `nice` 。由于 /usr 目录有可能在系统启动过程的早期不可用(比如尚未挂载)，所以这些二进制程序需要放置在根分区上：

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.14.2. Coreutils 的内容

安装的程序： basename, cat, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, shred, sleep, sort, split, stat, stty, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, yes

简要描述

`basename`	去掉文件名中的目录和后缀
`cat`	把文本文件的内容发送到标准输出
`chgrp`	改变文件和目录属组，属组可以使用组名或者组识别号表示
`chmod`	改变文件和目录的权限，权限可以使用符号或者八进制两种表达方式
`chown`	改变文件和目录的所有权(包括用户和/或组)
`chroot`	使用特定的目录作为执行某个命令或者交互 shell 的根目录(/)。在多数系统中，只有 root 用户能运行这个命令
`cksum`	输出指定的每个文件的CRC(循环冗余校验)校验和与字节数
`comm`	一行一行对两个已经排序的文件进行比较，在第三列中显示同一行是否相同
`cp`	复制文件
`csplit`	把一个文件按照给定的模式或者行号分成几块
`cut`	从指定的文件中提取特定的列送到标准输出
`date`	以特定的格式显示当前时间，或者设置系统日期
`dd`	以可选块长度复制文件，默认情况下从标准输入设备输出到标准输出设备。复制过程中，还可以对文件进行一些转换
`df`	显示参数中的文件所在分区磁盘空间的使用情况，如果没有给出文件参数就显示所有已经安装的文件系统的可用空间数量。
`dir`	列出给定目录的内容 (同 `ls` 命令)
`dircolors`	设置 LS_COLOR 环境变量(用来改变 `ls` 及相关工具默认颜色组合)
`dirname`	显示从文件名去掉非目录后缀之后的内容
`du`	显示参数使用的磁盘空间的数量，对于参数为目录还会列出每个子目录磁盘空间占用情况。
`echo`	显示给定字符串或变量值
`env`	在一个被修改的环境中运行一个程序
`expand`	把 tab 转换为空格符

expr	执行表达式计算
factor	输出所有指定整数的质因数
false	返回一个不成功或者逻辑假的结果
fmt	重新格式化指定文件的段落
fold	断开指定文件(默认是标准输入)较长的行，在屏幕上显示
groups	显示一个用户所在的组
head	显示每个指定文件的前几行(默认是10)
hostid	以16进制方式，显示当前主机的数字标志符
hostname	显示或设置主机名
id	显示某个用户或者当前用户的真实和有效的 UID、GID 。
install	复制文件，设置它们的权限，如果可能还设置拥有它们的用户和组
join	合并两个文件的行
link	创建从指定文件到指定名称的硬链接
ln	创建文件之间的硬/软(符号)连接
logname	显示当前用户的登录名
ls	列出指定目录的所有内容。缺省是将文件和子目录按字母顺序排列。
md5sum	显示或者校验 MD5 校验码。
mkdir	建立目录，使用给定的参数作为目录名。
mkfifo	以给定的参数作为名字建立FIFO(又叫"命名管道")文件。
mknod	使用给出的文件名，建立一个设备节点，也就是：FIFO、字符特殊文件(special file)或者块特殊文件(special file)。
mv	根据所给参数的不同，把文件或者目录移动到另外的目录或者将其改名
nice	修改某个进程的调度优先级
nl	把每个指定文件的内容写到标准输出，在每行加上行号
nohup	使某个命令不被挂起，并将输出重定向到一个日志文件。
od	以数字方式显示指定文件的内容，默认为八进制。
paste	将字段连接在一起，在字段之间自动插入分割符，默认的分割符是 Tab 。
pathchk	检查文件名是否是有效的或者是可移植的
pinky	一个轻量级的 finger 客户端，用来得到某个用户的信息。
pr	将文件分成适当大小的页送到打印机
printenv	显示环境变量
printf	根据给定的参数格式化输出数据，与 C 语言中的该函数相似。

`ptx`	为指定的文件提供一个排序索引
`pwd`	显示当前工作目录
`readlink`	显示指定符号链接的值
`rm`	删除文件或者目录
`rmdir`	删除目录(目录必需为空)
`seq`	以指定的步长输出一个数列
`sha1sum`	显示或校验 160 位的 SHA1 校验码
`shred`	安全删除一个文件，重写其占用的磁盘空间，使其无法恢复。
`sleep`	延迟一段时间
`sort`	对文件进行排序
`split`	把文件分成固定大小(字节或行数)的片断
`stat`	显示文件或者文件系统的状态
`stty`	改变和显示终端行的设置
`sum`	显示指定文件的校验和及块数
`sync`	刷新文件系统缓冲区，使磁盘和内存的数据同步。
`tac`	逆向显示指定的文件，最后一行在最前。
`tail`	显示每个指定文件的最后几行(默认是10)。
`tee`	从标准输入读取数据，输出到标准输出和指定的文件。
`test`	检查文件类型，以及进行变量的比较。
`touch`	把参数指定的文件的访问和修改时间改为当前的时间。如果文件不存在，它就建立一个空文件。
`tr`	从标准输入读入正文，对字符进行转换、压缩或者删除，然后写到标准输出
`true`	返回一个成功或者逻辑真的结果
`tsort`	对给定的文件进行拓扑排序
`tty`	显示标准输出设备连接终端的文件名
`uname`	打印系统信息
`unexpand`	把空格符转换成 tab
`uniq`	抛弃指定文件或者标准输入中内容重复的行
`unlink`	删除指定文件
`users`	显示在当前主机登录的用户名
`vdir`	同 **ls -l**

`wc`	统计文件中包含的字节数、单词数、行数
`who`	显示有哪些用户登录
`whoami`	打印当前用户的有效用户标志符
`yes`	重复输出"y"字符，直到被杀死。

6.15. Iana-Etc-2.10

Iana-Etc 软件包提供了网络服务和协议的数据。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 2.1 MB

6.15.1. 安装 Iana-Etc

下面的命令将 IANA 的原始数据转换为 `/etc/protocols` 和 `/etc/services` 数据文件能够识别的格式：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.15.2. Iana-Etc 的内容

安装的文件： /etc/protocols, /etc/services

简要描述

`/etc/protocols`	描述 TCP/IP 子系统可用的各种 Internet 协议。
`/etc/services`	将 internet 服务映射到一个包含端口号和所使用协议的文本名称。

6.16. M4-1.4.4

M4 软件包包含一个宏处理器。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 3 MB

6.16.1. 安装 M4

为编译 M4 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行：**make check** 。

安装软件包：

```
make install
```

6.16.2. M4 的内容

安装的程序： m4

简要描述

`m4`	M4 能够将宏展开并将输入拷贝到输出。宏可以是内嵌的也可以是用户定义的，并且可以接受很多参数。除了展开宏， `m4` 还有其它内置的功能，比如包含引用文件、运行 Unix 命令、进行整数运算、文本操作、循环等等。 `m4` 可以被用作一个编译器的前端或作为自身的一个宏处理程序。

6.17. Bison-2.2

Bison 软件包包括一个语法分析程序生成器。

预计编译时间： 0.6 SBU所需磁盘空间： 11.9 MB

6.17.1. 安装 Bison

为编译 Bison 做准备：

```
./configure --prefix=/usr
```

如果 `bison` 程序不在 $PATH 中的话，编译时将会出现缺乏国际化支持的错误信息。下面处理可以解决这个问题：

```
echo '#define YYENABLE_NLS 1' >> config.h
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

6.17.2. Bison 的内容

安装的程序： bison, yacc安装的库： liby.a

简要描述

`bison`	根据一系列规则来生成一个可以分析文本文件的结构的程序的程序，Bison 是一个替代 Yacc (Yet Another Compiler Compiler) 的语法分析程序生成器。
`yacc`	一个 `bison` 的包装，意思是程序仍然调用 `yacc` 而不是 `bison` ，它用 *-y* 选项调用 `bison` 。
`liby.a`	acc 库包含与 Yacc 兼容的 `yyerror` 和 `main` 函数，这个库通常不是很有用，但是 POSIX 需要它。

6.18. Ncurses-5.5

Ncurses 程序包提供字符终端处理库，包括面板和菜单。

预计编译时间： 0.7 SBU所需磁盘空间： 31 MB

6.18.1. 安装 Ncurses

Ncurses-5.5 有一个内存泄漏和一些显示的bug被发现，并修正了。应用这些修正：

```
patch -Np1 -i ../ncurses-5.5-fixes-1.patch
```

为编译 Ncurses 做准备：

```
./configure --prefix=/usr --with-shared --without-debug --enable-widec
```

配置选项的含义：

`--enable-widec`

这个选项导致宽字符库（例如， `libncursesw.so.5.5` ）将会替换正常的（例如， `libncurses.so.5.5` ）。这些宽字符库可以在 多字节和传统的8位 locale 下使用，然而正常的库一般只能在 8位 的locale环境下工作。宽字符和正常的库是源码兼容的，但不是二进制兼容的。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

赋予 ncurses 库文件可执行权限：

```
chmod -v 755 /usr/lib/*.5.5
```

修正一个不应该有可执行权限的库文件：

```
chmod -v 644 /usr/lib/libncurses++w.a
```

把库文件移到更合理的 `/lib` 目录里：

```
mv -v /usr/lib/libncursesw.so.5* /lib
```

由于库文件移动了，所以有的符号链接就指向了不存在的文件。需要重新创建这些符号链接：

```
ln -sfv ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

许多的程序依然希望链接器能够发现非宽字符的 Ncurses 库。通过符号链接和链接器脚本来欺骗它使其链接宽字符的库：

```
for lib in curses ncurses form panel menu ; do \
    rm -vf /usr/lib/lib${lib}.so ; \
    echo "INPUT(-l${lib}w)" >/usr/lib/lib${lib}.so ; \
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a ; \
done &&
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

最后，确保一些在编译的时候寻找 `-lcurses` 的老程序仍然可以编译：

```
echo "INPUT(-lncursesw)" >/usr/lib/libcursesw.so &&
ln -sfv libncurses.so /usr/lib/libcurses.so &&
ln -sfv libncursesw.a /usr/lib/libcursesw.a &&
ln -sfv libncurses.a /usr/lib/libcurses.a
```

注意

上面的说明并没有创建非宽字符的 Ncurses 库，因为没有软件包编译后在运行时需要链接到它们的。如果你因为一些只有二进制文件的程序，必须安装这样的库的话，按照下面的命令进行编译：

```
make distclean &&
./configure --prefix=/usr --with-shared --without-normal \
  --without-debug --without-cxx-binding &&
make sources libs &&
cp -av lib/lib*.so.5* /usr/lib
```

6.18.2. Ncurses 的内容

安装的程序： captoinfo(→tic), clear, infocmp, infotocap(→tic), reset(→tset), tack, tic, toe, tput, tset安装的库： libcursesw.{a,so} (symlink and linker script to libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} and

their non-wide-character counterparts without "w" in the library names.

简要描述

`captoinfo`	将 termcap 描述转化成 terminfo 描述
`clear`	如果可能，就进行清屏操作
`infocmp`	比较或显示 terminfo 描述
`infotocap`	将 terminfo 描述转化成 termcat 描述
`reset`	重新初始化终端到默认值
`tack`	terminfo 动作检测器。主要用来测试 terminfo 数据库中某一条目的正确性。
`tic`	Tic 是 terminfo 项说明的编译器。这个程序通过 ncurses 库将源代码格式的 terminfo 文件转换成编译后格式(二进制)的文件。 Terminfo 文件包含终端能力的信息。
`toe`	列出所有可用的终端类型，分别列出名称和描述。
`tput`	利用 terminfo 数据库使与终端相关的能力和信息值对 shell 可用，初始化和重新设置终端，或返回所要求终端为类型的长名。
`tset`	可以用来初始化终端
`libcurses`	链接到 `libncurses`
`libncurses`	用来在显示器上显示文本的库。一个例子就是在内核的 `make menuconfig` 进程中。
`libform`	在 ncurses 中使用表格
`libmenu`	在 ncurses 中使用菜单
`libpanel`	在 ncurses 中使用面板

6.19. Procps-3.2.6

Procps 包含有用于监视系统进程的程序。

预计编译时间： 0.1 SBU所需磁盘空间： 2.3 MB

6.19.1. 安装 Procps

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.19.2. Procps 的内容

安装的程序： free, kill, pgrep, pkill, pmap, ps, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w, watch安装的库： libproc.so

简要描述

`free`	报告系统中的空闲和已用内存数量(同时包括物理内存和交换内存)
`kill`	向进程发送消息
`pgrep`	基于进程名及其属性来查找进程
`pkill`	依据进程名及其属性向进程发送消息
`pmap`	报告所告所给定进程的内存映射
`ps`	显示当前正运行的进程列表
`skill`	向符合所给标准的进程发送消息
`slabtop`	实时显示内核 slap 缓冲的详细信息
`snice`	改变符合所给标准的进程的调度优先权
`sysctl`	在运行期间修改内核参数
`tload`	打印当前系统平均负荷曲线图
`top`	显示使用 CPU 最密集的进程列表，它提供了对实时的处理器行为的实时察看。
`uptime`	报告系统运行了多久，有多少用户登录，以及系统平均负荷。
`vmstat`	报告虚拟内存统计，并给出有关处进程、内存、块输入/输出(IO)、陷阱、CPU使用率。
`w`	显示哪个用户登录，在哪里以及何时登录的。
`watch`	重复运行所给的命令，以显示输出的第一次满屏，这将允许用户察看随时间变化的输出。
`libproc`	含有该软件包中大多数程序所需使用的函数

6.20. Sed-4.1.5

Sed 是一个流编辑器。

预计编译时间： 0.1 SBU所需磁盘空间： 6.4 MB

6.20.1. 安装 Sed

为编译 Sed 做准备：

```
./configure --prefix=/usr --bindir=/bin --enable-html
```

新配置选项的含义：

--enable-html

This builds the HTML documentation.

编译软件包：

```
make
```

要测试结果，请运行： **make check** 。

安装软件包：

```
make install
```

6.20.2. Sed 的内容

安装的程序： sed

简要描述

sed	流式(单向)过滤和改变文本文件

6.21. Libtool-1.5.22

GNU libtool 是一个通用库支持脚本，将使用动态库的复杂性隐藏在统一的、可移植的接口中。

预计编译时间： 0.1 SBU所需磁盘空间： 16.6 MB

6.21.1. 安装 Libtool

为编译 Libtool 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

6.21.2. Libtool 的内容

安装的程序： libtool, libtoolize安装的库： libltdl.{a,so}

简要描述

`libtool`	提供通用的库编译支持。
`libtoolize`	提供了一种标准方式来将 `libtool` 支持加入到一个软件包中
`libltdl`	隐藏 dlopening 库的复杂细节

6.22. Perl-5.8.8

Perl 将 C, sed, awk 和 sh 的最佳特性集于一身，是一种强大的编程语言。

预计编译时间： 1.5 SBU所需磁盘空间： 143 MB

6.22.1. 安装 Perl

为了运行测试套件，要先创建一个基本的 `/etc/hosts` 文件，好几个测试都需要它来解析 localhost 的名称：

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

对 Perl 的设置进行更多的控制，你可以运行交互的 `Configure` 脚本，精心选择编译配置。如果你能接受 Perl 的自动配置(这是很明智的)，就用下面的命令：

```
./configure.gnu --prefix=/usr \
    -Dman1dir=/usr/share/man/man1 \
    -Dman3dir=/usr/share/man/man3 \
    -Dpager="/usr/bin/less -isR"
```

配置选项的含义：

`-Dpager="/usr/bin/less -isR"`

纠正 `perldoc` 代码调用 `less` 程序时的一个错误。

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`

因为 Groff 还没有安装， `Configure` 会认为我们不想安装 Perl 的 man 手册。应用这个参数来改变这种情况：

编译软件包：

```
make
```

要测试结果，请运行： **`make test`** 。

安装软件包：

```
make install
```

6.22.2. Perl 的内容

安装的程序： a2p, c2ph, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, libnetcfg, perl, perl5.8.8(→perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, psed(→s2p), pstruct(→c2ph), s2p, splain, xsubpp安装的库：太多了，有好几百个，无法在这里全部列出！

简要描述

`a2p`	把 awk 翻译成 Perl
`c2ph`	显示 `cc -g -S` 产生的 C 语言结构。
`dprofpp`	显示 Perl 的 profile 数据。
`enc2xs`	为 Encode 模块编译 Perl 扩展，用于 Unicode 字符映射或 Tcl 编码文件。
`find2perl`	将 `find` 命令翻译成 Perl 代码。
`h2ph`	将 `.h` 的C头文件转成 `.ph` 的perl头文件
`h2xs`	将 `.h` 的 C 头文件转成 perl 程序扩展
`instmodsh`	一个监测安装 Perl 模块的 Shell 脚本，甚至可以从已安装模块中创建压缩包。
`libnetcfg`	可以用来配置 `libnet`
`perl`	综合了 C, `sed`, `awk`, `sh` 特性和能力于一体的强大的编程语言
`perl5.8.8`	`perl` 的硬连接
`perlbug`	生成关于 perl 和相关模块的 bug 报告，并且 mail 给他们。
`perlcc`	从 perl 程序生成可执行文件
`perldoc`	显示嵌于 perl 安装目录或者一个 perl 脚本的 .pod 格式的小文档。
`perlivp`	Perl 安装验证过程，可以用它来验证 Perl 及其库是否安装正常。
`piconv`	A 是 Perl 版本的字符编码转换程序，类似于 `iconv`
`pl2pm`	将 Perl4 样式的 `.pl` 库文件转化为 Perl5 样式的 `.pm` 库模块的工具
`pod2html`	将 pod 格式的文件转为 html 格式
`pod2latex`	将 pod 格式的文件转为 LaTeX 格式
`pod2man`	将 pod 数据转为格式化的 *roff 输入
`pod2text`	将 pod 数据转为格式化的 ASCII 文本
`pod2usage`	打印文件内嵌的 pod 文档的使用信息
`podchecker`	检查 pod 格式的文档的语法
`podselect`	有选择的打印 pod 文档内容到标准输出
`psed`	是 Perl 版本的流式编辑器，类似于 `sed`
`pstruct`	显示 `cc -g -S` 产生的 C 语言结构
`s2p`	把 `sed` 脚本翻译成 Perl 脚本
`splain`	强制 Perl 输出冗余警告信息
`xsubpp`	把 Perl XS 代码转换成 C 代码

6.23. Readline-5.1

Readline 软件包是一个提供命令行编辑和历史纪录功能的库集合。

预计编译时间： 0.1 SBU所需磁盘空间： 10.2 MB

6.23.1. 安装 Readline

上游开发者已经修正了自从 Readline-5.1 之后版本的一些问题。应用这些修正：

```
patch -Np1 -i ../readline-5.1-fixes-3.patch
```

重新安装 Readline 会将老的库libraryname重命名为<libraryname>.old。然而着并不是一个问题。在某些情况下它会引发 `ldconfig` 的一个链接bug。应用下面的两个sed命令可以避免这种情况：

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

为编译 Readline 做准备：

```
./configure --prefix=/usr --libdir=/lib
```

编译软件包：

```
make SHLIB_LIBS=-lncurses
```

make 选项的含义：

`SHLIB_LIBS=-lncurses`

这个选项强制 Readline 链接到 `libncurses` 库。

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

给 Readline 动态库更多恰当的权限：

```
chmod -v 755 /lib/lib{readline,history}.so*
```

将静态库移动到一个更合理的位置：

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

删除 `/lib` 中的 `.so` 文件，并将它们重新连接到 `/usr/lib` 中：

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.5 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

6.23.2. Readline 的内容

安装的库： libhistory.{a,so}, libreadline.{a,so}

简要描述

`libhistory`	提供一个统一的调用历史行的用户接口
`libreadline`	应用于各种需要命令行接口的应用程序的统一的用户接口的辅助程序

6.24. Zlib-1.2.3

Zlib 软件包包含 zlib 库，很多程序中的压缩或者解压缩程序都会用到这个库。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 3.1 MB

6.24.1. 安装 Zlib

注意

如果在环境变量中指定了 `CFLAGS` 的话，Zlib 就不能正常编译共享库。如果你想使用自定义的 `CFLAGS` 环境变量，请在下述整个 configure 命令的过程中始终把 `-fPIC` 指令加在 `CFLAGS` 的最前面，结束后还必须再撤销它。

为编译 Zlib 做准备：

```
./configure --prefix=/usr --shared --libdir=/lib
```

编译软件包：

```
make
```

要测试结果，请运行：**`make check`** 。

安装共享库：

```
make install
```

上面的命令将会在 `/lib` 目录下安装一个 `.so` 文件。我们将要移除它并重新连接到 `/usr/lib` 目录下：

```
rm -v /lib/libz.so
ln -sfv ../../lib/libz.so.1.2.3 /usr/lib/libz.so
```

编译静态库(非共享库)：

```
make clean
./configure --prefix=/usr
make
```

要测试静态库可以用这个命令：**`make check`** 。

安装静态库：

```
make install
```

修正静态库的权限：

```
chmod -v 644 /usr/lib/libz.a
```

6.24.2. Zlib 的内容

安装的库： libz.{a,so}

简要描述

`libz`	包含很多程序都用到的压缩和解压函数

6.25. Autoconf-2.59

Autoconf 能生成用于自动配置源代码的 shell 脚本

预计编译时间： 少于 0.1 SBU所需磁盘空间： 7.2 MB

6.25.1. 安装 Autoconf

为编译 Autoconf 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。这可能要花费比较长的时间，大约 3 SUB。另外，因为要用到 Automake 的原因，跳过测试二。为了全面测试，可以在 Auotomake 安装完后重新测试。

安装软件包：

```
make install
```

6.25.2. Autoconf 的内容

安装的程序： autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, ifnames

简要描述

`autoconf`	一个产生可以自动配置源代码包，生成shell脚本的工具，以适应各种类UNIX 系统的需要。 `autoconf` 产生的配置脚本在运行时独立于 `autoconf` ，因此使用这些脚本的用户不需要安装 `autoconf` 。
`autoheader`	能够创建供 configure 脚本使用的 C *#define* 语句模板文件。
`autom4te`	一个 M4 宏处理器的包装
`autoreconf`	当 `autoconf` 和 `automake` 的模版文件被改变的时候，以正确的顺序自动运行 `autoconf` , `autoheader` , `aclocal` , span> `automake` , `gettextize` , `libtoolize` 以节约时间。
`autoscan`	为软件包创建 `configure.in` 文件。它以命令行参数中指定的目录为根(如果未给定参数则以当前目录为根)的目录树中检查源文件，搜索其中的可移植性问题，为那个软件包创建一个 `configure.scan` 文件以充当一个预备性的 `configure.in` 文件。
`autoupdate`	将 `configure.in` 文件中 `autoconf` 宏的旧名称更新为当前名称
`ifnames`	为一个软件包写 `configure.in` 文件提供帮助，它打印软件包中那些在 C 预处理器中已经使用了的标识符。如果一个包已经设置成具有某些可移植属性，这个程序能够帮助指出它的 `configure` 脚本应该如何检查。它可以用来填补由 `configure.in` 产生的 `autoscan` 中的隔阂。

6.26. Automake-1.9.6

Automake 与 Autoconf 配合使用，产生 Makefile 文件。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 7.9 MB

6.26.1. 安装 Automake

为编译 Automake 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。 This takes a long time, about 10 SUB.

安装软件包：

```
make install
```

6.26.2. Automake 的内容

安装的程序： acinstall, aclocal, aclocal-1.9.6, automake, automake-1.9.6, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, ylwrap

简要描述

`acinstall`	用来安装 aclocal 风格的 M4文件的脚本
`aclocal`	根据 `configure.in` 文件的内容，自动生成 `aclocal.m4` 文件
`aclocal-1.9.6`	`aclocal` 的硬链接
`automake`	根据 `Makefile.am` 文件的内容，自动生成 `Makefile.in` 文件。在目录的顶层运行该命令，可以为一个包建立所有的 `Makefile.in` 文件。通过扫描 `configure.in` 文件，它可以自动找到每一个合适的 `Makefile.am` 文件并且产生相应的 `Makefile.in` 文件。
`automake-1.9.6`	`automake` 的一个硬链接
`compile`	包装了编译器的脚本
`config.guess`	用来为特定的 build, host, target 尝试猜测标准的系统名称的脚本
`config.sub`	配置验证子脚本
`depcomp`	在编译程序的同时产生其依赖信息的脚本
`elisp-comp`	按字节编译 Emacs Lisp 代码
`install-sh`	能安装程序、脚本、数据文件的脚本
`mdate-sh`	打印程序和目录更改时间的脚本
`missing`	一个用来填充在安装过程检查出的缺失的 GNU 程序空位的脚本
`mkinstalldirs`	产生目录树结构的脚本
`py-compile`	编译 Python 程序
`symlink-tree`	为整个目录创建符号链接的脚本
`ylwrap`	包装了 `lex` 和 `yacc` 的脚本

6.27. Bash-3.1

Bash 是 Bourne-Again Shell 的缩写，它在 UNIX 系统中作为 shell 被广泛使用。

预计编译时间： 0.4 SBU所需磁盘空间： 25.8 MB

6.27.1. 安装 Bash

如果你下载了Bash的文档包，可以通过下面的命令安装：

```
tar -xvf ../bash-doc-3.1.tar.gz &&
sed -i "s|htmldir = @htmldir@|htmldir = /usr/share/doc/bash-3.1|" \
    Makefile.in
```

上游开发者解决了从Bash-3.1后的一些问题：

```
patch -Np1 -i ../bash-3.1-fixes-8.patch
```

为编译 Bash 做准备：

```
./configure --prefix=/usr --bindir=/bin \
    --without-bash-malloc --with-installed-readline
```

配置选项的含义：

`--with-installed-readline`

这个选项是告诉 Bash 使用已经安装的系统的 `readline` 库，而不是它自己的 readline 版本。

编译软件包：

```
make
```

要测试结果，请运行： **`make tests`** 。

安装软件包：

```
make install
```

运行新编译的 `bash` 程序来替换正在执行的这一个：

```
exec /bin/bash --login +h
```

注意

上面命令里使用的参数指示 `bash` 产生一个交互式的登陆 shell 并且继续禁用哈希功能，以便新的程序一旦被安装就立即投入使用。

6.27.2. Bash 的内容

安装的程序： bash, bashbug, sh(→bash)

简要描述

`bash`	作为命令行解释器被广泛使用。它能在执行命令前解释非常复杂的命令行参数，这使它成为一个强大的工具。
`bashbug`	帮助用户用标准格式编写和提交有关 `bash` 的bug报告的脚本。
`sh`	指向 `bash` 的符号连接。当运行 `sh` 的时候，`bash` 会尽量模仿老的 `sh` 历史环境来运行，同时遵循 POSIX 标准。

6.28. Bzip2-1.0.3

Bzip2 包含了对文件进行压缩和解压缩的工具，对于文本文件，`bzip2` 比传统的 `gzip` 拥有更高压缩比。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 5.3 MB

6.28.1. 安装 Bzip2

下面的补丁可以为这个软件包安装相应的文档：

```
patch -Np1 -i ../bzip2-1.0.3-install_docs-1.patch
```

`bzgrep` 命令并不将传递给它的文件名中的 '|' 和 '&' 进行转义，这就会允许别有用心的用户执行任意命令。下面的补丁可以解决这个问题：

```
patch -Np1 -i ../bzip2-1.0.3-bzgrep_security-1.patch
```

`bzdiff` 脚本仍然会使用原来的 `tempfile` 程序。可以使用 `mktemp` 来替换：

```
sed -i 's@tempfile -d /tmp -p bz@mktemp -p /tmp@' bzdiff
```

为编译 Bzip2 做准备：

```
make -f Makefile-libbz2_so
make clean
```

make 参数的含义：

`-f Makefile-libbz2_so`

这会采用一个另外一个 `Makefile` 来编译 Bzip2，也就是这里的 Makefile-libbz2_so 文件，它创建一个动态链接库 `libbz2.so` ，然后把Bzip2的工具都链接到这个库上。

编译并测试软件包：

```
make
```

如果重新安装Bzip2，必须首先执行 **`rm -vf /usr/bin/bz*`** ，否则下面的 `make install` 会出错。

安装Bzip2：

```
make install
```

把 `bzip2` 二进制共享库拷贝到 `/bin` 目录，创建必要的符号链接，再做一些清理工作：

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.28.2. Bzip2 的内容

安装的程序： bunzip2(→bzip2), bzcat(→bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless, bzmore安装的库： libbz2.{a,so}

简要描述

`bunzip2`	解压使用 bzip2 压缩的文件
`bzcat`	解压缩指定的文件到标准输出
`bzcmp`	对 bzip2 压缩的文件运行 `cmp` 命令
`bzdiff`	对 bzip2 压缩的文件运行 `diff` 命令
`bzgrep`	对 bzip2 压缩的文件运行 `grep` 命令
`bzegrep`	对 bzip2 压缩的文件运行 `egrep` 命令
`bzfgrep`	对 bzip2 压缩的文件运行 `fgrep` 命令
`bzip2`	使用 Burrows-Wheeler 块排列文本压缩算法和霍夫曼编码来压缩文件。压缩比要大于 gzip 工具使用的基于"Lempel-Ziv"的压缩算法(如 `gzip` 格式)，接近 PPM 统计压缩算法族的压缩比。
`bzip2recover`	试图从被破坏的 bzip2 文件中恢复数据
`bzless`	对 bzip2 压缩的文件运行 `less` 命令
`bzmore`	对 bzip2 压缩的文件运行 `more` 命令
`libbz2*`	利用 Burrows-Wheeler 算法，实现无损块顺序数据压缩的库文件。

6.29. Diffutils-2.8.1

Diffutils 软件包里的程序向你显示两个文件或目录的差异，常用来生成软件的补丁。

预计编译时间： 0.1 SBU所需磁盘空间： 6.3 MB

6.29.1. 安装 Diffutils

POSIX 要求 `diff` 命令能够根据当前的locale处理 whitespace（空白符）。 下面的patch可以解决这个问题：

```
patch -Np1 -i ../diffutils-2.8.1-i18n-1.patch
```

上面的这个patch将会导致用一个无效的程序`help2man`来重新编译 `diff.1` man 帮助。结果导致 `diff` 的 man 不可读。我们可以通过改变 `man/diff.1` 的时间戳来避免这个问题：

```
touch man/diff.1
```

为编译 Diffutils 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.29.2. Diffutils 的内容

安装的程序： cmp, diff, diff3, sdiff

简要描述

`cmp`	比较两个文件，并指出它们是否不同及不同的字节。
`diff`	比较两个文件或目录，并指出哪些文件的哪些行不同。
`diff3`	逐行比较三个文件
`sdiff`	合并两个文件，并以交互方式输出结果

6.30. E2fsprogs-1.39

E2fsprogs 提供用于 `ext2` 文件系统的工具。它还支持 `ext3` 日志文件系统。

预计编译时间： 0.4 SBU所需磁盘空间： 31.2 MB

6.30.1. 安装 E2fsprogs

推荐在 E2fsprogs 的源码目录外面来编译它：

```
mkdir -v build
cd build
```

为编译 E2fsprogs 做准备：

```
../configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs --disable-evms
```

配置选项的含义：

`--with-root-prefix=""`

有的程序(如 `e2fsck`)对系统来说是非常重要的，例如，在 `/usr` 没有挂载的情况下。这些程序和库就应放在像 `/lib` 和 `/sbin` 这些目录中。如果没有把上面的参数传递给 E2fsprogs 的 configure 脚本，它就会把程序放在 `/usr` 目录下。

`--enable-elf-shlibs`

这会创建共享的库，供 E2fsprogs 包中的一些程序使用。

`--disable-evms`

这个选项禁止了企业卷管理系统(EVMS)插件的支持。因为这个插件并没有更新到适合最新的 EVMS 接口并且 EVMS 并不是基本 LFS 系统的一部分，所以我们并不需要这个插件。请参考 EVMS 网站 *http://evms.sourceforge.net/* 以获得更多信息。

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

E2fsprogs 的一个测试会尝试分配 256 MB 内存。如果你没有充足的RAM空间，推荐打开足够的交换分区。参见节 2.3, "在新分区上创建文件系统" 和 节 2.4, "挂载新分区" 获取关于创建和激活交换分区的细节。

安装二进制文件和文档：

```
make install
```

安装共享库：

```
make install-libs
```

6.30.2. E2fsprogs 的内容

安装的程序： badblocks, blkid, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, filefrag, findfs, fsck, fsck.ext2, fsck.ext3, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs, uuidgen.安装的库： libblkid.{a,so}, libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libss.{a,so}, libuuid.{a,so}

简要描述

`badblocks`	用来检查设备(通常是硬盘分区)上的坏块
`blkid`	定位并打印出块设备属性的命令行工具
`chattr`	在 `ext2` 和 `ext3` 文件系统上改变文件属性
`compile_et`	用来将错误代码(error-code)和相关出错信息的列表 转化为适用于 `com_err` 库的 C 语言文件
`debugfs`	文件系统调试器。能用来检查和改变 `ext2` 文件系统的状态
`dumpe2fs`	打印特定设备上现存的文件系统的超级块(super block)和块群(blocks group)的信息
`e2fsck`	用来检查和修复 `ext2` 和 `ext3` 文件系统
`e2image`	将关键的 `ext2` 文件系统数据保存到一个文件中
`e2label`	显示或者改变指定设备上的 `ext2` 文件系统标识
`filefrag`	报告一个文件的碎片情况
`findfs`	通过卷标或通用唯一标识符(UUID)寻找文件系统
`fsck`	用来检查或者修理文件系统
`fsck.ext2`	默认检查 `ext2` 文件系统

fsck.ext3	默认检查 ext3 文件系统
logsave	把一个命令的输出保存在日志文件中
lsattr	列出 ext2 文件系统上的文件属性
mk_cmds	将一个包含命令列表的文件转化为适用于子系统库 libss 的 C 源文件
mke2fs	用来创建 ext2 或 ext3 文件系统
mkfs.ext2	默认创建 ext2 文件系统
mkfs.ext3	默认创建 ext3 文件系统
mklost+found	在 ext2 文件系统上创建一个 lost+found 目录，并给该目录预分配磁盘数据块，以减轻 e2fsck 命令的负担。
resize2fs	可以用来增大或缩小 ext2 文件系统
tune2fs	调整 ext2 文件系统的可调参数
uuidgen	创建一个新的通用唯一标识符(UUID)。这个新 UUID 可以被认为是在所有已创建的 UUID 中独一无二的，不论是在本地的系统或者别的系统，过去还是将来。
libblkid	包含设备识别和节点释放的库函数
libcom_err	通用错误显示库
libe2p	用于 dumpe2fs , chattr , lsattr
libext2fs	允许用户级的程序操作 ext2 文件系统
libss	用于 debugfs
libuuid	用来给对象产生通用唯一标识符(UUID)使之可以在本地系统之外引用

- 后退

  Diffutils-2.8.1

- 前进

  File-4.17

- 上一级
- 首页.

6.31. File-4.17

File 是用来判断文件类型的工具。

预计编译时间： 0.1 SBU所需磁盘空间： 7.5 MB

6.31.1. 安装 File

为编译 File 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.31.2. File 的内容

安装的程序： file安装的库： libmagic.{a,so}

简要描述

`file`	测试每一个指定的文件并且试图对它们进行分类。有三个测试集，按照下面的顺序执行：文件系统测试、幻数(magic number)测试、语言测试。第一个测试成功后会打印文件文件类型。
`libmagic`	包含产生幻数(magic number)的函数，供 `file` 程序使用。

6.32. Findutils-4.2.27

Findutils 包含查找文件的工具，既能即时查找(递归的搜索目录，并可以显示、创建和维护文件)，也能在数据库里查找(通常比递归查找快但是在数据库没有及时更新的情况下，结果并不可靠)。

预计编译时间： 0.2 SBU所需磁盘空间： 12 MB

6.32.1. 安装 Findutils

为编译 Findutils 做准备：

```
./configure --prefix=/usr --libexecdir=/usr/lib/findutils \
    --localstatedir=/var/lib/locate
```

配置选项的含义：

--localstatedir

将 `locate` 数据库的位置指定为 `/var/lib/locate` ，以符合 FHS 标准。

编译软件包：

```
make
```

要测试结果，请运行： **make check** 。

安装软件包：

```
make install
```

LFS-Bootscripts 包中的一些脚本依赖于 `find` 。因为在系统启动的前期， `/usr` 目录还是无法访问的（比如还没有挂载上），因此这个程序需要放在根分区上。 `updatedb` 脚本也需要用完全路径来修正：

```
mv -v /usr/bin/find /bin
sed -i -e 's/find:=${BINDIR}/find:=\/bin/' /usr/bin/updatedb
```

6.32.2. Findutils 的内容

安装的程序： bigram, code, find, frcode, locate, updatedb, xargs

简要描述

`bigram`	以前用来创建 `locate` 数据库。
`code`	以前用来创建 `locate` 数据库。它是 `frcode` 的前身。
`find`	在一个目录和其子目录里面找符合条件的文件
`frcode`	被 `updatedb` 调用来压缩文件名列表，它使用的是前端压缩(front-compression)，可以减小数据库4到5倍。
`locate`	扫描一个文件名称数据库，可以列出在数据库中符合条件的文件或者目录。
`updatedb`	更新 `locate` 数据库。它会扫描整个文件系统，包括所有挂载的文件系统(除非设定参数禁止)，并且把每一个找到的文件和目录放到 `locate` 数据库里面。
`xargs`	可以在一系列文件上运行同一个命令

6.33. Flex-2.5.33

Flex 软件包包含一个能生成识别文本模式的程序的工具。

预计编译时间： 0.1 SBU所需磁盘空间： 8.4 MB

6.33.1. 安装 Flex

为编译 Flex 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

一些程序试图在 `/usr/lib` 下面寻找 `lex` 库，可以创建一个符号链接来满足要求：

```
ln -sv libfl.a /usr/lib/libl.a
```

一些程序并不知道 `flex` 而是试图寻找 `lex` 程序(flex 是实现 lex 功能的另一种也是更好的选择)。为了满足少数一些程序的需要，我们将创建一个 `lex` 脚本，这个脚本调用 `flex` 并通过它来模仿 `lex` 的输出文件命名惯例：

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

6.33.2. Flex 的内容

安装的程序： flex, lex安装的库： libfl.a

简要描述

`flex`	是一个用来生成识别文本的模式程序的工具。模式识别在很多程序中是非常有用，用户设置一些查询规则，然后 flex 可以生成一个查询程序。人们使用 flex 可以比亲自编写查询程序更便捷。
`lex`	在 `lex` 仿真模式下运行 `flex` 的一个脚本
`libfl.a`	`flex` 库

6.34. GRUB-0.97

GRUB 程序包包含 GRand 统一引导装载程序。

预计编译时间： 0.2 SBU所需磁盘空间： 10.2 MB

6.34.1. 安装 GRUB

如果你把这个包缺省的优化参数(包括 `-march` 和 `-mcpu` 参数)改变的话，它会有些不正常的表现。因此，如果你定义了任何优化参数的话，比如 CFLAGS 和 CXXFLAGS ，我们劝你在编译时 unset 或修改它们。

开始先打一个补丁来达到更好的硬件侦测、修复 GCC 4.x 的一些问题以及为一些磁盘控制器提供更好的 SATA 支持：

```
patch -Np1 -i ../grub-0.97-disk_geometry-1.patch
```

为编译 GRUB 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
mkdir -v /boot/grub
cp -v /usr/lib/grub/i386-pc/stage{1,2} /boot/grub
```

把 i386-pc 换成适合你的平台的路径。

i386-pc 目录还包含一些 *stage1_5 文件，是为不同的文件系统准备的。看看有哪些文件，并把你所需要的拷贝到 /boot/grub 目录下。多数人需要 e2fs_stage1_5 和/或 reiserfs_stage1_5 文件。

6.34.2. GRUB 的内容

安装的程序： grub, grub-install, grub-md5-crypt, grub-set-default, grub-terminfo, mbchk

简要描述

`grub`	GRUB 的命令解释 shell
`grub-install`	在指定设备上安装 GRUB
`grub-md5-crypt`	以 MD5 加密一个密码
`grub-set-default`	为 Grub 设置默认启动入口
`grub-terminfo`	从 terminfo 名称产生 terminfo 命令。如果你在一个不常见的终端时，可以使用这个命令。
`mbchk`	检查多重启动内核的格式

6.35. Gawk-3.1.5

Gawk 软件包包含用于管理文本文件的程序。

预计编译时间： 0.2 SBU所需磁盘空间： 18.2 MB

6.35.1. 安装 Gawk

在某些情况下，Gawk-3.1.5会释放一块没有分配的内存。应用下面的patch可以解决问题：

```
patch -Np1 -i ../gawk-3.1.5-segfault_fix-1.patch
```

为编译 Gawk 做准备：

```
./configure --prefix=/usr --libexecdir=/usr/lib
```

由于在 `configure` 脚本中的一个 bug ，Gawk 就不会发现 Glibc 中的某些方面的locale支持。 这个bug会导致很多问题。例如，Gettext 测试单元会失败。解决这个问题的方法就是在 `config.h` 中添加丢失的宏定义：

```
cat >>config.h <<"EOF"
#define HAVE_LANGINFO_CODESET 1
#define HAVE_LC_MESSAGES 1
EOF
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

6.35.2. Gawk 的内容

安装的程序： awk(→gawk), gawk, gawk-3.1.5, grcat, igawk, pgawk, pgawk-3.1.5, pwcat

简要描述

`awk`	指向 `gawk` 的链接
`gawk`	`awk` 的GNU版本，用来管理文本文件的程序。
`gawk-3.1.5`	`gawk` 的硬链接
`grcat`	读取组数据库 `/etc/group`
`igawk`	赋予 `gawk` 包含文件的能力
`pgawk`	`gawk` 的概要分析(profiling)版本
`pgawk-3.1.5`	`pgawk` 的硬链接
`pwcat`	`/etc/passwd` 读取密码数据库

6.36. Gettext-0.14.5

Gettext 用于系统的国际化和本地化，可以在编译程序的时候使用本国语言支持(NLS)，可以使程序的输出使用用户设置的语言而不是英文。

预计编译时间： 1 SBU所需磁盘空间： 65 MB

6.36.1. 安装 Gettext

为编译 Gettext 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。这需要大约 5 SBU 。

安装软件包：

```
make install
```

6.36.2. Gettext 的内容

安装的程序： autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, xgettext安装的库： libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so}, libgettextsrc.so

简要描述

`autopoint`	将标准的 gettext infrastructure 文件拷贝到源码包中
`config.charset`	产生一个依赖于系统的字符编码表
`config.rpath`	输出依赖于系统的变量，描述如何在可执行程序中设置库文件的实时查找路径。
`envsubst`	替换 shell 格式化字符串中的环境变量

`gettext`	通过在消息列表中查找，来将一种自然语言的消息翻译成用户的本地语言
`gettext.sh`	主要作为一个shell的函数库来为 gettext 服务
`gettextize`	拷贝所有的标准 Gettext 文件到软件包的顶层目录下，以进行国际化。
`hostname`	用不同的格式显示一个网络主机名
`msgattrib`	根据翻译目录中消息的属性过滤它们，并且操作这些属性。
`msgcat`	将给定的 `.po` 文件合并到一起
`msgcmp`	比较两个 `.po` 文件，看它们是否包含相同的 msgid 字符串
`msgcomm`	找出不同 `.po` 文件中的相同信息
`msgconv`	把一个翻译列表转化成另一种字符编码
`msgen`	建立一个英文翻译目录
`msgexec`	对一个翻译列表中的所有翻译执行同一个命令
`msgfilter`	对一个翻译列表中的所有翻译使用同一个过滤器
`msgfmt`	从翻译列表生成一个二进制的消息列表
`msggrep`	将一个翻译列表中的所有符合给定格式或者属于给定源文件的消息展开
`msginit`	生成一个新的 `.po` 文件，并使用用户环境中的消息来对这个文件进行初始化。
`msgmerge`	将两个翻译合并成一个
`msgunfmt`	将二进制翻译文件反编译成源文件
`msguniq`	将翻译目录中重复的翻译合并成一个
`ngettext`	显示依赖于数字格式的文本文件的本国语言翻译
`xgettext`	从源文件中展开消息行, 用来创建起始翻译模板
`libasprintf`	定义 *autosprintf* 类，使 C 的函数以 C++ 程序可使用的结构输出，与 *<string>* 字符串和 *<iostream>* 流一起使用。
`libgettextlib`	包含多个 gettext 程序使用的函数，是私有库
`libgettextpo`	用来写处理 `.po` 文件的程序。当 Gettext 自带的标准程序(如 `msgcomm`, `msgcmp`, `msgattrib`, `msgen`)不能满足要求时，可以使用这个库。
`libgettextsrc`	包含多个 gettext 程序使用的函数，是私有库。

6.37. Grep-2.5.1a

Grep 可以搜索文件中符合指定匹配模式的行。

预计编译时间： 0.1 SBU所需磁盘空间： 4.8 MB

6.37.1. 安装 Grep

当前的 Grep 包有很多bug,尤其是对多字节的loacles的支持。RedHat 采用下面的这个patch来解决部分问题：

```
patch -Np1 -i ../grep-2.5.1a-redhat_fixes-2.patch
```

需要修改测试文件的权限，才能使打过patch后在测试中通过：

```
chmod +x tests/fmbtest.sh
```

为编译 Grep 做准备：

```
./configure --prefix=/usr --bindir=/bin
```

编译软件包：

```
make
```

要测试结果，请运行： `make check` 。

安装软件包：

```
make install
```

6.37.2. Grep 的内容

安装的程序： egrep(→grep), fgrep(→grep), grep

简要描述

`egrep`	打印出匹配扩展正则表达式模式的行
`fgrep`	对固定字符串列表进行匹配
`grep`	对基本正则表达式进行匹配

6.38. Groff-1.18.1.1

Groff 软件包包含几个处理和格式化文本的程序。Groff 把标准的文本和特殊的命令翻译成格式化的输出，就像你在 man 手册页里看到的那样。

预计编译时间： 0.4 SBU所需磁盘空间： 39.2 MB

6.38.1. 安装 Groff

应用下面的patch来添加 "ascii8" 和 "nippon" 设备到 Groff：

```
patch -Np1 -i ../groff-1.18.1.1-debian_fixes-1.patch
```

注意

这些设备在转换一些非 ISO-8859-1 编码的非英语的 man 手册页时，会用到。现在对于 Groff-1.19.x 没有同样功能的patch：

许多屏幕字体没有 Unicode 的单引号和破折号。告诉 Groff 使用等价的 ASCII 字符：

```
sed -i -e 's/2010/002D/' -e 's/2212/002D/' \
    -e 's/2018/0060/' -e 's/2019/0027/' font/devutf8/R.proto
```

Groff 希望环境变量 `PAGE` 包含缺省的纸张尺寸。对于在美国的人来说，应当使用 `PAGE=letter` ，如果你住在其他地方，可能需要把 `PAGE=letter` 改成 `PAGE=A4` 。

为编译 Groff 做准备：

```
PAGE=<paper_size> ./configure --prefix=/usr --enable-multibyte
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

有些文档程序，比如 `xman` ，没有下面的符号链接会不能正常工作：

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.38.2. Groff 的内容

安装的程序： addftinfo, afmtodit, eqn, eqn2graph, geqn(→eqn), grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl(→tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, refer, soelim, tbl, tfmtodit, troff

简要描述

`addftinfo`	读取一个 troff 字体文件并增加一些 `groff` 系统使用的附加点阵字体。
`afmtodit`	建立同 `groff` 和 `grops` 一起使用的字体文件。
`eqn`	将嵌于 troff 输入文件中的方程描述翻译成 `troff` 能够理解的命令
`eqn2graph`	把 EQN 等式转换成反图像
`geqn`	指向 `eqn` 的连接
`grn`	`groff` 的 gremlin 文件预处理器
`grodvi`	`groff` 的产生 TeX dvi 格式的驱动
`groff`	groff 文档格式系统的前端。通常它调用 `troff` 程序和对指定设备适用的后处理器
`groffer`	把 groff 文件和 man 文档显示在 X 和 tty 上
`grog`	读取文件然后猜测使用 *-e* ，*-man* ，*-me* ，*-mm* ，*-ms* ，*-p* ，*-s* ，*-t* 中的哪个 `groff` 参数来打印文件。并把带有这个参数的 `groff` 命令输出到标准输出。
`grolbp`	LBP-4 和 LBP-8 激光打印机系列的 `groff` 驱动
`grolj4`	产生适用于HP Laserjet4 打印机的 PCL5 格式的 `groff` 驱动
`grops`	将 GNU `troff` 的输出翻译成 PostScript
`grotty`	将 GNU `troff` 的标准输出翻译成适合类打字机设备的格式
`gtbl`	指向 `tbl` 的连接
`hpftodit`	使用 `groff -Tlj4` 从一个 HP-tagged 字体文件中创建 groff 使用的字体文件。
`indxbib`	建立同 `refer`，`lookbib`，`lkbib` 一起使用的文件的文献数据库的反向列表
`lkbib`	在文献数据库中搜索包括指定关键字的条目，并将其输出到标准输出
	打印一个标准错误的提示，除非标准输入不是终端。从标准输入读入关

`lookbib`	键字搜索在指定文件中的文献数据库中的含有这些关键字的条目，并将结果输出到标准输出。
`mmroff`	一个简易的 `groff` 预处理器
`neqn`	将方程格式化，使其成为适应 ASCII 输出的脚本
`nroff`	这个脚本用 `nroff` 命令仿真 `groff` 命令
`pfbtops`	将 `.pfb` 格式的 Postscript 字体翻译成 ASCII 码
`pic`	将内嵌于 troff 或 TeX 输入文件中的图像编译成 `troff` 或 TeX 理解的指令
`pic2graph`	把 PIC 图表转换成反图像
`post-grohtml`	将GNU `troff` 的输出翻译成 HTML
`pre-grohtml`	将GNU `troff` 的输出翻译成 HTML
`refer`	将一个文件拷贝到标准输出并丢弃 *.[* 和 *.]* 之间作为引用的内容和在 *.R1* 和 *.R2* 之间解释如何处理这些引用的命令
`soelim`	读取文件将其中的 *.so* 文件 表格替换为 文件 的内容
`tbl`	将内嵌于 troff 或者 TeX 输入文件中的表格编译成 `troff` 或者 TeX 理解的指令
`tfmtodit`	建立 `groff -Tdvi` 使用的字体文件
`troff`	和 Unix 的 `troff` 高度兼容。一般运行 `groff` 来调用它，`groff` 依照合适的顺序并使用合适的参数来执行预处理程序和后处理程序。

6.39. Gzip-1.3.5

gzip 包含用 Lempel-Ziv 编码(LZ77)来压缩和解压文件的程序。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 2.2 MB

6.39.1. 安装 Gzip

Gzip 有两个安全漏洞，下面的补丁可以修正它：

```
patch -Np1 -i ../gzip-1.3.5-security_fixes-1.patch
```

为编译 Gzip 做准备：

```
./configure --prefix=/usr
```

`gzexe` 脚本里包含对 `gzip` 程序的硬路径引用。由于我们后面要改变 `gzip` 程序的位置，就需要用下面的命令改变脚本中的硬路径：

```
sed -i 's@"BINDIR"@/bin@g' gzexe.in
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

把 `gzip` 程序移动到 `/bin` 目录并创建一些常用的符号连接：

```
mv -v /usr/bin/gzip /bin
rm -v /usr/bin/{gunzip,zcat}
ln -sv gzip /bin/gunzip
ln -sv gzip /bin/zcat
ln -sv gzip /bin/compress
ln -sv gunzip /bin/uncompress
```

6.39.2. Gzip 的内容

安装的程序： compress(→gzip), gunzip(→gzip), gzexe, gzip, uncompress(→gunzip), zcat(→gzip), zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, znew

简要描述

`compress`	压缩和解压缩文件
`gunzip`	解压由 gzip 压缩过的文件
`gzexe`	将文件压缩成可以自解压的可执行文件
`gzip`	通过 Lempel-Ziv 编码(LZ77)压缩指定的文件
`uncompress`	解压由 gzip 压缩过的文件
`zcat`	将解压缩的数据写到标准输出上
`zcmp`	在压缩文件上调用 `cmp` 命令
`zdiff`	在压缩文件上调用 `diff` 命令
`zegrep`	在压缩文件上调用 `egrep` 命令
`zfgrep`	在压缩文件上调用 `fgrep` 命令
`zforce`	强制性地为每一个 gzip 文件加上 `.gz` 扩展名，这样 `gzip` 就不会对它们再次进行压缩。这个程序可能在一个文件经过传输后名字被截短的情况下能够派上用场。
`zgrep`	在压缩文件上调用 `grep` 命令
`zless`	在压缩文件上调用 `less` 命令
`zmore`	在压缩文件上调用 `more` 命令
`znew`	将 `.z` 格式的文件(使用 compress 压缩)转压缩成 `.gz` 格式(使用 gzip 压缩)

6.40. Inetutils-1.4.2

Inetutils 包含基本的网络程序。

预计编译时间： 0.2 SBU所需磁盘空间： 8.9 MB

6.40.1. 安装 Inetutils

应用一个patch，使其能够在GCC-4.0.3下编译：

```
patch -Np1 -i ../inetutils-1.4.2-gcc4_fixes-3.patch
```

我们并不安装 Inetutils 的全部程序，然而，它默认会把所有程序的 man 文档都装上。下面的补丁能解决这个问题：

```
patch -Np1 -i ../inetutils-1.4.2-no_server_man_pages-1.patch
```

为编译 Inetutils 做准备：

```
./configure --prefix=/usr --libexecdir=/usr/sbin \
    --sysconfdir=/etc --localstatedir=/var \
    --disable-logger --disable-syslogd \
    --disable-whois --disable-servers
```

配置选项的含义：

`--disable-logger`

阻止 inetutils 安装 `logger` 程序，脚本利用这个程序向系统日志守护进程传递消息。我们不安装它是因为 Util-linux 包含一个更好的版本。

`--disable-syslogd`

这个参数阻止 inetutils 安装 System Log Daemon(系统日志守护进程)，我们将在后面的 Sysklogd 软件包中安装它。

`--disable-whois`

阻止 inetutils 编译 `whois` 客户端，因为它已经很陈旧了。在 BLFS book 里面有安装更好的 `whois` 客户端的指导。

`--disable-servers`

阻止安装几种网络服务器。这些服务器对于基本的 LFS 系统是不合适的，有的还不安全，很多服务器都有更好的替代者。参见 *http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html* 。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

把 `ping` 程序移动到符合 FHS 标准的位置：

```
mv -v /usr/bin/ping /bin
```

6.40.2. Inetutils 的内容

安装的程序： ftp, ping, rcp, rlogin, rsh, talk, telnet, tftp

简要描述

`ftp`	文件传输协议程序
`ping`	向网络主机发送请求应答包，并报告回复所需的时间。
`rcp`	远程文件拷贝
`rlogin`	远程登陆
`rsh`	运行远程 shell
`talk`	与另一个用户交谈
`telnet`	TELNET 协议接口
`tftp`	小文件传输程序

6.41. IPRoute2-2.6.16-060323

IPRoute2 包含了基本的和高级的基于 IPv4 网络的程序。

预计编译时间： 0.2 SBU所需磁盘空间： 4.8 MB

6.41.1. 安装 IPRoute2

编译软件包：

```
make SBINDIR=/sbin
```

make 选项的含义：

`SBINDIR=/sbin`

确保将 IPRoute2 包中的二进制文件安装到 `/sbin` 目录中以符合 FHS 标准，因为一些 IPRoute2 二进制文件将会被 LFS-Bootscripts 使用。

这个软件包没有附带测试程序。

安装软件包：

```
make SBINDIR=/sbin install
```

`arpd` 二进制文件链接到在 `/usr` 目录中的Berkeley DB库，并且使用数据库 `/var/lib/arpd/arpd.db` 。因此，按照 FHS，它必须存在于 `/usr/sbin` 目录中。移动它到那里：

```
mv -v /sbin/arpd /usr/sbin
```

6.41.2. IPRoute2 的内容

安装的程序： arpd, ctstat(→lnstat), ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat(→lnstat), ss, tc.

简要描述

`arpd`	用户空间的 ARP 守护进程。用在大型网络中，那里内核空间的 ARP 实现不是很合适;或者是用在设置一个蜜罐。
`ctstat`	连接状态工具
`ifcfg`	`ip` 命令的shell脚本包装
`ifstat`	显示网络接口的统计信息，包括接口发送和接收到的包数量。
`ip`	主可执行程序，它包含以下几个功能：`ip link _ [device] _` 查看和修改设备状态 `ip addr` 查看地址的特性，添加新地址、删除旧地址。`ip neighbor` 查看邻居的特性，添加新邻居、删除旧邻居。`ip rule` 查看和修改路由规则 `ip route` 查看路由表和修改路由表规则 `ip tunnel` 查看和修改 IP 隧道及其特性 `ip maddr` 查看和修改多播地址及其特性 `ip mroute` 设置、修改、删除多播路由 `ip monitor` 不间断的监视设备状态、地址、路由
`lnstat`	提供 Linux 网络统计信息，用于替代旧的 `rtstat` 程序。
`nstat`	显示网络统计信息
`routef`	`ip route` 的一个组件，用于刷新路由表。
`routel`	`ip route` 的一个组件，用于列出路由表。
`rtacct`	显示 `/proc/net/rt_acct` 文件的内容
`rtmon`	路由监视工具
`rtpr`	将 `ip -o` 的输出转换为可读的格式
`rtstat`	路由状态工具
`ss`	类似于 `netstat` 命令，显示活动的连接。
`tc`	流量控制，用于实现服务质量(QOS)和服务级别(COS)：`tc qdisc` 建立排队规则 `tc class` 建立基于级别的队列调度 `tc estimator` 估算网络流量 `tc filter` 设置 QOS/COS 包过滤器 `tc policy` 设置 QOS/COS 规则

6.42. Kbd-1.12

Kbd 包含键盘映射表和键盘工具。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 12.3 MB

6.42.1. 安装 Kbd

Backspace 键和 Delete 键的功能在 kbd 包的键盘映射中是不一样的。下面的 patch 修正了 i386 的键盘映射中的这个问题：

```
patch -Np1 -i ../kbd-1.12-backspace-1.patch
```

打完 patch 之后，Backspace 键会产生字符编码 127，Delete 键会产生一个著名的逃脱序列。

应用 patch 来修正 Kbd 中的 `setfont` 在 GCC-4.0.3 下编译出错的问题：

```
patch -Np1 -i ../kbd-1.12-gcc4_fixes-1.patch
```

为编译 Kbd 做准备：

```
./configure --datadir=/lib/kbd
```

配置选项的含义：

`--datadir=/lib/kbd`

这个选项把键盘布局信息存放到根分区内，而不是存放在默认的 `/usr/share/kbd` 。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

注意

对于一些语言（例如，白俄罗斯），在 Kbd 包中没有提供相应的键盘映射。系统假定使用 ISO-8859-5 编码，通常使用 CP1251 键盘映射。 这些语种的用户需要单独下载相应的键盘映射：

LFS-Bootscripts 包中的一些脚本依赖于 `kbd_mode` ， `openvt` ，和 `setfont` 。因为 `/usr` 在启动的早些时候是无法访问的（没有挂载）。那些二进制文件需要放在根分区上：

```
mv -v /usr/bin/{kbd_mode,openvt,setfont} /bin
```

6.42.2. Kbd 的内容

安装的程序： chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable(→psfxtable), psfgettable(→psfxtable), psfstriptable(→psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, showconsolefont, showkey, unicode_start, unicode_stop

简要描述

chvt	改变前台虚拟终端
deallocvt	重新分配不用的虚拟终端
dumpkeys	显示键盘转换表
fgconsole	显示活动虚拟控制台的数量
getkeycodes	显示内核中扫描码与键盘码的转换表
kbd_mode	设置或显示键盘模式
kbdrate	设置或显示键盘重复和延迟的速度
loadkeys	加载键盘转换表
loadunimap	加载内核的 Unicode 到字体(unicode-to-font)之间的影射表
mapscrn	把用户定义的输出字符影射表加载到控制台驱动器中。注意这个程序已经过时，它实现的功能已经并入 setfont 程序。
openvt	在一个新虚拟终端启动一个程序
psfaddtable	链接到 psfxtable
psfgettable	链接到 psfxtable
psfstriptable	链接到 psfxtable
psfxtable	一套处理控制台字体的 Unicode 字符表的工具
resizecons	让内核改变控制台的大小
setfont	改变控制台的 EGA 或 VGA 字体
setkeycodes	告诉内核的键盘驱动程序在扫描码/键码(scancode-to-keycode)影射表中加入新的影射，当你的键盘上有某些特殊建的时候这个就很有用了。
setleds	设置当前终端键盘的发光二极管(LED)标志
setmetamode	设置键盘的元键(meta key)
showconsolefont	显示当前 EGA / VGA 终端的屏幕字体
showkey	测试键盘发出的扫描码和键码
unicode_start	使控制台进入 UNICODE 模式。在 LFS 系统中从不使用，因为应用程序并未配置为支持 UNICODE 。
unicode_stop	终止控制台的 UNICODE 模式

6.43. Less-394

Less 软件包包含一个文本文件查看器。

预计编译时间： 0.1 SBU所需磁盘空间： 2.6 MB

6.43.1. 安装 Less

为编译 Less 做准备：

```
./configure --prefix=/usr --sysconfdir=/etc
```

配置选项的含义：

`--sysconfdir=/etc`

这个选项告诉程序建立软件包时在 `/etc` 目录里查找配制文件。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.43.2. Less 的内容

安装的程序： less, lessecho, lesskey

简要描述

`less`	一个文件显示或分页程序。它显示某文件的内容，并可以分卷、寻找字符串、跳转到某一标记。
`lessecho`	在 Unix 系统中，可以用来扩展元字符，比如 * 和 ?
`lesskey`	为 `less` 定义按健

6.44. Make-3.80

Make 自动地确定一个大型程序的哪些片段需要重新编译，并且发出命令去重新编译它们。

预计编译时间： 0.1 SBU所需磁盘空间： 7.8 MB

6.44.1. 安装 Make

为编译 Make 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

6.44.2. Make 的内容

安装的程序： make

简要描述

`make`	自动地确定一个大型程序的哪些片段需要重新编译，并且发出命令去重新编译它们。

6.45. Man-DB-2.4.3

Man-DB 包含查找和显示 man 手册页的程序。

预计编译时间： 0.2 SBU所需磁盘空间： 9 MB

6.45.1. 安装 Man-DB

对 Man-DB 的源码需要做三个调整。

第一，改变已翻译的 Man-DB 带的 manual 页的位置，使其在传统和 UTF-8 的 locale 环境下均能使用：

```
mv man/de{_DE.88591,} &&
mv man/es{_ES.88591,} &&
mv man/it{_IT.88591,} &&
mv man/ja{_JP.eucJP,} &&
sed -i 's,\*_\*,??,' man/Makefile.in
```

第二，使用 `sed` 的替换功能来删除 `man_db.conf` 文件中的 "/usr/man" 行， 来避免在使用像 `whatis` 这样的命令时的冗余结果：

```
sed -i '/\t\/usr\/man/d' src/man_db.conf.in
```

第三，改变 Man-DB 在运行时应该能够发现的程序的记录，但它们还没有安装：

```
cat >>include/manconfig.h.in <<"EOF"
#define WEB_BROWSER "exec /usr/bin/lynx"
#define COL "/usr/bin/col"
#define VGRIND "/usr/bin/vgrind"
#define GRAP "/usr/bin/grap"
EOF
```

`col` 是 Util-linux 的一部分， `lynx` 基于文本的浏览器（参见 BLFS 的安装说明）, `vgrind` 把程序源码转换为 Groff 的输入， `grap` 在 Groff 文档中对图表排版非常有用。 `vgrind` 和 `grap` 不是查看 manual 页所必须的。它们不是 LFS 的一部分，也不是 BLFS 的一部分，但是在完成 LFS 之后，你应该能够自己安装。

为编译 Man-DB 做准备：

```
./configure --prefix=/usr --enable-mb-groff --disable-setuid
```

配置选项的含义：

`--enable-mb-groff`

通知 `man` 在格式化非ISO-8859-1格式的 manual 页时，使用 "ascii8" 和 "nippon" Groff 设备。

`--disable-setuid`

使 `man` 不能给用户 `man` 设置uid位。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

一些软件包提供这个版本 `man` 无法显示的 UTF-8 编码的 man 手册页。 下面的脚本将会允许它们中的一些转换成为下面表中的编码格式。Man-DB 期望 manual 页的编码格式是下面表中的一项。在显示它们的时 候，可以根据需要转换成实际的 locale 编码，因此它们能够在传统的和 UTF-8 的locale下显示。因为脚本是在系统构建和对公共数据 中限制使用的，所以，我们不必担心错误检测和使用非预期的临时文件名。

```
cat >>convert-mans <<"EOF"
#!/bin/sh -e
FROM="$1"
TO="$2"
shift ; shift
while [ $# -gt 0 ]
do
        FILE="$1"
        shift
        iconv -f "$FROM" -t "$TO" "$FILE" >.tmp.iconv
        mv .tmp.iconv "$FILE"
done
EOF
install -m755 convert-mans  /usr/bin
```

一些关于 man 和 info 页的压缩可以在 BLFS 手册中找到（*http://www.linuxfromscratch.org/blfs/view/cvs/postlfs/compressdoc.html*）。

6.45.2. Non-English Manual Pages in LFS

每一种 Linux 的发行版在存储 manual 页的字符编码方面都有自己的方法。比如，RedHat 以 UTF-8 编码存储所有的 manual 页;然而 Debian 使用不同的语言编码(通常是8位的)。这 就导致包在为不同发行版设置 manual 页时不兼容。

LFS采用跟 Debian 一样的方法。这是因为 Man-DB 不能够理解 UTF-8 存储的 man 手册页。Man-DB 比 Man 好在在任意的 locale 下不需要额外的配置就可以使用。最后，我们没有选用 RedHat 的方法，因为它的 `groff` 文本格式组织的不好。

语言跟字符编码的关联已经列在下面的表中。Man-DB 会在查看的时候自动的将它们转变为 locale 的编码。

Table 6.1. manual 页的字符编码表 pages

Language (code)	Encoding
Danish (da)	ISO-8859-1
German (de)	ISO-8859-1
English (en)	ISO-8859-1
Spanish (es)	ISO-8859-1
Finnish (fi)	ISO-8859-1
French (fr)	ISO-8859-1
Irish (ga)	ISO-8859-1
Galician (gl)	ISO-8859-1
Indonesian (id)	ISO-8859-1
Icelandic (is)	ISO-8859-1
Italian (it)	ISO-8859-1
Dutch (nl)	ISO-8859-1
Norwegian (no)	ISO-8859-1
Portuguese (pt)	ISO-8859-1
Swedish (sv)	ISO-8859-1
Czech (cs)	ISO-8859-2
Croatian (hr)	ISO-8859-2
Hungarian (hu)	ISO-8859-2
Japanese (ja)	EUC-JP
Korean (ko)	EUC-KR
Polish (pl)	ISO-8859-2
Russian (ru)	KOI8-R
Slovak (sk)	ISO-8859-2
Turkish (tr)	ISO-8859-9

注意

不在上面列表中的语言的 Manual 页是不支持的。挪威语无法使用，因为从 no_NO 到 nb_NO 的locale转变。韩语因为不完整的 patch 也是功能不全的。 of the incomplete Groff patch.

如果 manual 页的编码是 Man-DB 所预期的，manual 页就会被拷贝到 `/usr/share/man/<language code>` 。例如，法语的 manual 页(*http://ccb.club.fr/man/man-fr-1.58.0.tar.bz2*) 可以使用下面的命令安装：

```
mkdir -p /usr/share/man/fr &&
cp -rv man? /usr/share/man/fr
```

如果 manual 页是 UTF-8 编码的（例如，针对 "RedHat" 的） ，而不是上面列表中的。它们不得不在安装之前从 UTF-8 转变为列表中的编码。这可以通过 `convert-mans` 来实现。比如，西班牙的 manual 页 (*http://ditec.um.es/~piernas/manpages-es/man-pages-es-1.55.tar.bz2*) 可以通过下面的命令来安装：

```
mv man7/iso_8859-7.7{,X}
convert-mans UTF-8 ISO-8859-1 man?/*.?
mv man7/iso_8859-7.7{X,}
make install
```

注意

在转换的过程中需要除去 `man7/iso_8859-7.7` 文件，因为它已经存在于ISO-8859-1中，这是 man-pages-es-1.55 的一个bug。下一个版本可能就不会需要了。

6.45.3. Man-DB 的内容

安装的程序： accessdb, apropos, catman, convert-mans,lexgrog, man, mandb, manpath, whatis, zsoelim

简要描述

`accessdb`	将 `whatis` 数据库的内容转储为人类可读形式
`apropos`	搜索 `whatis` 数据库，显示包含给定字符串的系统命令的简短描述。
`catman`	创建或更新预格式化的 manual 页
`convert-mans`	重新格式化 man 手册页，以使 Man-DB 能够显示它们
`lexgrog`	显示一行给定 manual 页的摘要信息。
`man`	格式化并显示请求的 manual 页
`mandb`	创建或更新 `whatis` 数据库
`manpath`	显示 $MANPATH 的内容或在man.conf中设置的搜索路径（如果 $MANPATH 没有设置），以及用户的环境变量。
`whatis`	搜索 `whatis` 数据库，显示包含给定关键字的系统命令的简短描述。
`zsoelim`	读取文件并用提到的 *file* 的内容来替换 *.so file* 格式的行。

6.46. Mktemp-1.5

Mktemp 软件包包含用于在 shell 脚本中创建安全临时文件的程序。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 0.4 MB

6.46.1. 安装 Mktemp

许多脚本目前仍然使用被反对使用的类似于 `mktemp` 的 `tempfile` 程序，我们现在要给 Mktemp 打一个补丁，以使它包含 `tempfile` 包装：

```
patch -Np1 -i ../mktemp-1.5-add_tempfile-3.patch
```

为编译 Mktemp 做准备：

```
./configure --prefix=/usr --with-libc
```

配置选项的含义：

`--with-libc`

这个使得 `mktemp` 程序从系统的 C 库中使用 *mkstemp* 和 *mkdtemp* 的功能。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
make install-tempfile
```

6.46.2. Mktemp 的内容

安装的程序： mktemp, tempfile

简要描述

`mktemp`	使用安全性较强的方式创建临时文件，用于脚本中。
`tempfile`	使用比 `mktemp` 安全性较弱的方式创建临时文件，但是能够满足向后的兼容性。

6.47. Module-Init-Tools-3.2.2

Module-Init-Tools 包含处理 2.5.47 及以上版本的内核模块时使用的工具。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 7 MB

6.47.1. 安装 Module-Init-Tools

首先更正一个当模块被指定使用正则表达式时会出现的潜在问题：

```
patch -Np1 -i ../module-init-tools-3.2.2-modprobe-1.patch
```

执行下面的命令进行测试（注意 `make distclean` 命令需要清理源码树，因为作为测试过程的一部分，源码会重新编译：

```
./configure &&
make check &&
make distclean
```

为编译 Module-Init-Tools 做准备：

```
./configure --prefix=/ --enable-zlib
```

编译软件包：

```
make
```

安装软件包：

```
make INSTALL=install install
```

make 参数的含义：

`INSTALL=install`

正常情况下，如果二进制文件已经存在了，`make install` 就不会安装它们。 这个选项是调用 `install` 而不是使用默认封装的脚本。

6.47.2. Module-Init-Tools 的内容

安装的程序： depmod, generate-modprobe.conf, insmod, insmod.static, lsmod, modinfo, modprobe, rmmod

简要描述

`depmod`	创建一个可加载内核模块的依赖关系文件， `modprobe` 用它来自动加载模块。
`generate-modprobe.conf`	从一个现存的2.2 或者 2.4版本内核的模块设置中创建一个 modprobe.conf 文件
`insmod`	向正在运行的内核加载模块
`insmod.static`	`insmod` 的静态编译版本
`lsmod`	显示当前已加载的内核模块信息
`modinfo`	检查与内核模块相关联的目标文件，并打印出所有能得到的信息。
`modprobe`	利用 `depmod` 创建的依赖关系文件来自动加载相关的模块
`rmmod`	从当前运行的内核中卸载模块

6.48. Patch-2.5.4

Patch 根据"补丁"文件的内容来修改原来的文件。补丁文件通常是用 `diff` 程序创建的，包含如何修改文件的指导。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 1.6 MB

6.48.1. 安装 Patch

为编译 Patch 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.48.2. Patch 的内容

安装的程序： patch

简要描述

`patch`	根据一个 `patch` 文件来对文件进行修改。通常情况下，一个 patch 文件是一个差别清单。这个清单用 `diff` 程序创建。通过将这些差别应用到原始文件，patch 创建出修订版本。

6.49. Psmisc-22.2

Psmisc 包含有用于显示进程信息的程序。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 2.2 MB

6.49.1. 安装 Psmisc

为编译 Psmisc 做准备：

```
./configure --prefix=/usr --exec-prefix=""
```

配置选项的含义：

`--exec-prefix=""`

这个确保 Psmisc 二进制文件按照 FHS 标准被安装在 `/bin` 而不是 `/usr/bin` ，因为一些 Psmisc 二进制文件将被 LFS-Bootscripts 使用。

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

没有理由把 `pstree` 和 `pstree.x11` 程序安装在 `/bin` 中，所以将他们移动到 `/usr/bin` 中：

```
mv -v /bin/pstree* /usr/bin
```

默认情况下， Psmisc 的 `pidof` 程序未被安装。 这通常情况下不是问题。因为它将在这之后的 Sysvinit 包中被安装，而且这个包提供了一个更好的 `pidof` 程序。如果你打算不使用 Sysvinit ，则可通过创建下面的符号连接来安装完整的 Psmisc ：

```
ln -sv killall /bin/pidof
```

6.49.2. Psmisc 的内容

安装的程序： fuser, killall, pstree, pstree.x11(→pstree)

简要描述

`fuser`	报告使用所给文件或文件系统的进程的进程ID(PID)。
`killall`	通过进程名来终止进程，它发送消息到所有正在运行任意所给指令的进程。
`oldfuser`	报告使用所给文件或文件系统的进程的进程ID(PID)。
`pstree`	以目录树的形式显示所有正在运行的进程
`pstree.x11`	同 `pstree` ，只是它在退出前要求确认

6.50. Shadow-4.0.15

Shadow 包含用于在安全方式下处理密码的程序。

预计编译时间： 0.3 SBU所需磁盘空间： 18.6 MB

6.50.1. 安装 Shadow

注意

如果你打算强制使用高强度密码，请参考 *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* 以获得如何在安装 Shadow 之前先安装 Cracklib 的指导。然后在下面的 `configure` 命令中加入 *`--with-libcrack`* 项。

为编译 Shadow 做准备：

```
./configure --libdir=/lib --enable-shared --without-selinux
```

配置选项的含义：

`--without-selinux`

selinux 的支持默认是打开的，但是 selinux 不包含在 LFS 基本系统中。如果这个选项不设置， `configure` 脚本会报错。

禁止安装 `groups` 程序和它的手册，Coreutils 软件包提供了一个更好的版本：

```
sed -i 's/groups$(EXEEXT) //' src/Makefile
find man -name Makefile -exec sed -i '/groups/d' {} \;
```

禁止安装中文和韩文的手册页，因为Man-DB不能很好的格式化它们：

```
sed -i -e 's/ ko//' -e 's/ zh_CN zh_TW//' man/Makefile
```

Shadow 支持在 UTF-8 编码环境下的其他编码的手册。Man-DB 可以通过我们安装的 `convert-mans` 脚本来转换并显示它们。

```
 for i in de es fi fr id it pt_BR; do
    convert-mans UTF-8 ISO-8859-1 man/${i}/*.?
done

for i in cs hu pl; do
    convert-mans UTF-8 ISO-8859-2 man/${i}/*.?
done

convert-mans UTF-8 EUC-JP man/ja/*.?
convert-mans UTF-8 KOI8-R man/ru/*.?
convert-mans UTF-8 ISO-8859-9 man/tr/*.?
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

Shadow 使用两个文件来设置系统的身份认证。下面安装这两个设置文件：

```
cp -v etc/{limits,login.access} /etc
```

不使用默认的 *crypt* 加密方法， 而使用更为安全的 *MD5* 对密码进行加密， 并且它同样允许密码长于 8 个字符。同时将用户邮箱位置从过时的 `/var/spool/mail` 修改到 `/var/mail` 是也有必要的(Shadow 普遍默认使用这个位置)。这两件事都可以通过在将相应的配置文件复制至目标位置之前，对其进行更改来达到：

```
sed -e's@#MD5_CRYPT_ENAB.no@MD5_CRYPT_ENAB yes@' \
    -e 's@/var/spool/mail@/var/mail@' \
    etc/login.defs > /etc/login.defs
```

注意

如果你在编译 Shadow 时启用了 Cracklib 支持，请在下面的 `sed` 命令中插入：

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
    /etc/login.defs
```

移动一些放错位置的符号连结或程序到正确位置：

```
mv -v /usr/bin/passwd /bin
```

移动 Shadow 的动态库到一个更为合适的地方：

```
mv -v /lib/libshadow.*a /usr/lib
rm -v /lib/libshadow.so
ln -sfv ../../lib/libshadow.so.0 /usr/lib/libshadow.so
```

`useradd` 程序的 `-D` 选项要求存在 `/etc/default` 目录以便程序能够正常工作：

```
mkdir -v /etc/default
```

6.50.2. 配置 Shadow

这个软件包中含有用来增加、修改和删除用户或组、设置和更改他们的密码、和执行其他管理任务的工具。为了获得对 *password shadowing (影子密码)* 的完全解释，请参见 `doc/HOWTO` 文件，它在解包后的源码目录树中。假如要使用 Shadow 支持，请注意那些需要对密码进行校验的程序(如显示管理器、FTP 程序、pop3进程等)必须兼容 Shadow 。也就是说，他们需要能够与影子密码一起工作。

为了使用影子密码，运行以下指令：

```
pwconv
```

为使用组影子密码，运行：

```
grpconv
```

6.50.3. Setting the root password

通过运行以下命令来设置 *root* 用户的密码：

```
passwd root
```

6.50.4. Shadow 的内容

安装的程序： chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg(→newgrp), su, useradd, userdel, usermod, vigr(→vipw), vipw安装的库： libshadow.{a,so}

简要描述

chage	用于设置必须对密码进行更改的最大间隔天数
chfn	用于对用户的全名及其他信息进行修改
chgpasswd	用于对一整个系列的组账号密码进行更新
chpasswd	用于对一整个系列的用户账号密码进行更新
chsh	用于更改一个用户的默认的登录 shell
expiry	检查并加强当前的密码过期策略
faillog	用于检查记录登录失败的日志，或是设置账户在被锁定前最大的登录失败次数，亦可用于重置登录失败的次数。
gpasswd	用来增加和删除组中的成员和管理员
groupadd	用指定的名称建一个组
groupdel	删除指定名称的组
groupmod	用来修改所指定组的名称或 GID
grpck	校验组文件 /etc/group 和 /etc/gshadow 的完整性
grpconv	从正常组文件中创建或更新一影子组文件
grpunconv	从 /etc/gshadow 更新 /etc/group 并将前者删除
lastlog	报告最近的所有用户的登录或是所指定用户的登录
login	被系统用来允许用户进行登录
logoutd	一个后台程序，用来加强对登录时间和端口进行限制
newgrp	用来在登录会话期间对当前的 GID 进行修改
newusers	用来对一整个系列的用户账户进行创建或更新
nologin	显示一个账户不可用的信息，被设计用来作为那些不准登录的账户的默认 shell。
passwd	用来对一个用户或组账户进行密码修改
pwck	校验密码文件 /etc/passwd 和 /etc/shadow 的完整性
pwconv	从一个正常的密码文件中创建或更新一影子密码文件
pwunconv	从 /etc/shadow 更新 /etc/passwd 并删除前者
sg	当一个用户的 GID 被设置到所给的组时执行所指定的命令
su	用另一个用户和组ID来运行一个 shell
useradd	用所给名称建立一个新的用户或更新默认新用户的信息
userdel	删除所指定的用户账户
usermod	用来更改所给用户的登录名、用户标识(UID)、shell、最初组、主目录等
vigr	编辑 /etc/group 或 /etc/gshadow 文件

`vipw`	编辑 `/etc/passwd` 或 `/etc/shadow` 文件
`libshadow`	包含了本包中大部分程序所用到的函数

6.51. Sysklogd-1.4.1

Sysklogd 包含记录系统日志信息的程序，比如内核处理意外事务的日志。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 0.6 MB

6.51.1. 安装 Sysklogd

下面的补丁修正了许多问题，包括在2.6系列的内核上编译 Sysklogd 会遇到的问题：

```
patch -Np1 -i ../sysklogd-1.4.1-fixes-1.patch
```

下面的patch使 sysklogd 逐字的对待日志信息中0x80--0x9f段的字符，而不是采用八进制进行替换。未打patch的sysklogd在UTF-8编码下会损害日志信息：

```
patch -Np1 -i ../sysklogd-1.4.1-8bit-1.patch
```

编译软件包：

```
make
```

这个软件包没有附带测试程序。

安装软件包：

```
make install
```

6.51.2. 配置 Sysklogd

创建一个新的 `/etc/syslog.conf` 文件：

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# End /etc/syslog.conf
EOF
```

6.51.3. Sysklogd 的内容

安装的程序： klogd, syslogd

简要描述

`klogd`	一个系统守护进程，截获并且记录下 LINUX 内核日志信息。
`syslogd`	记录下系统里所有提供日志记录的程序给出的日志和信息内容。每一个被记录的消息至少包含时间戳和主机名(通常还包括程序名)。

6.52. Sysvinit-2.86

Sysvinit 软件包包含一些控制系统启动、运行、关闭的程序。

预计编译时间： 少于 0.1 SBU所需磁盘空间： 1 MB

6.52.1. 安装 Sysvinit

当运行级别被改变(比如，正在关闭系统)，`init` 向那些由 `init` 自身开启的，并且将不会在新的运行级别里运行的线程发送终端信号。当 `init` 做上面这些事情时，会输出像"Sending processes the TERM signal"这样的信息，这看起来就像它正在向那些系统正在运行的程序发送上面这些信息一样。要避免错误地理解这个信息，可以修改源码以便可以代替为读起来像"Sending processes started by init the TERM signal"的信息，可以用下面命令：

```
sed -i 's@Sending processes@& started by init@g' \
    src/init.c
```

编译软件包：

```
make -C src
```

这个软件包没有附带测试程序。

安装软件包：

```
make -C src install
```

6.52.2. 配置 Sysvinit

运行下面命令，创建一个新的 `/etc/inittab` 文件：

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

6.52.3. Sysvinit 的内容

安装的程序： bootlogd, halt, init, killall5, last, lastb(→last), mesg, mountpoint, pidof(→killall5), poweroff(→halt), reboot(→halt), runlevel, shutdown, sulogin, telinit(→init), utmpdump, wall

简要描述

`bootlogd`	把启动信息记录到一个日志文件
`halt`	正常情况下等效于 `shutdown` 加上 *-h* 参数(当前系统运行级别是 0 时除外)。它将告诉内核去中止系统，并在系统正在关闭的过程中将日志记录到 `/var/log/wtmp` 文件里。
`init`	当内核已经初始化硬件，接管引导程序，开启指令线程时，init 会被第一个启动。
`killall5`	发送一个信号到所有进程，但那些在它自己设定级别的进程将不会被这个运行的脚本所中断。
`last`	给出哪一个用户最后一次登录(或退出登录)，它搜索 `/var/log/wtmp` 文件，出给出系统引导、关闭、运行级别改变等信息。
`lastb`	给出登失败的尝试，并写入日志 `/var/log/btmp`
`mesg`	控制是否允许其他用户也有向系统所有用户发送信息的权限
`mountpoint`	检查给定的目录是否是一个挂载点
`pidof`	报告给定程序的PID号
`poweroff`	告诉内核中止系统并且关闭系统(参见 `halt`)
`reboot`	告诉内核重启系统(参见 `halt`)
`runlevel`	告前一个和当前的系统运行级别，并且将最后一些运行级别写入 `/var/run/utmp`
`shutdown`	使系统安全关闭，向所有线程发送关闭信号并且通知所有已经登录的系统用户系统即将关闭。
`sulogin`	允许 *root* 登录，它通常情况下是在系统在单用户模式下运行时，由 `init` 所派生。
`telinit`	告诉 `init` 将切换到那一个运行级
`utmpdump`	以一个多用户友好的方式列出已经给出的登录文件的目录
`wall`	向所有已经登录的用户写入一个信息

6.53. Tar-1.15.1

Tar 包含一个归档程序。

预计编译时间： 0.2 SBU所需磁盘空间： 13.7 MB

6.53.1. 安装 Tar

应用一个patch来解决一些使用GCC-4.0.3时测试单元的问题：

```
patch -Np1 -i ../tar-1.15.1-gcc4_fix_tests-1.patch
```

当文件大于 4 GB 并且使用 `-S` 选项时，tar 有一个 bug 。下面的补丁修正了这个问题：

```
patch -Np1 -i ../tar-1.15.1-sparse_fix-1.patch
```

Tar最近版本都存在一个缓冲区溢出漏洞，应用下面的补丁修正这个问题：

```
patch -Np1 -i ../tar-1.15.1-security_fixes-1.patch
```

为编译 Tar 做准备：

```
./configure --prefix=/usr --bindir=/bin --libexecdir=/usr/sbin
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

6.53.2. Tar 的内容

安装的程序： rmt, tar

简要描述

`rmt`	通过一个 Internet 连接线程实施远程操作一个磁带驱动器
`tar`	从压缩档案里创建和解压文件，也可理解为压缩包(tarball)。

6.54. Texinfo-4.8

Texinfo 软件包包含读取、写入、转换 Info 文档的程序。

预计编译时间： 0.2 SBU所需磁盘空间： 16.6 MB

6.54.1. 安装 Texinfo

`info` 程序假定一个字符串在屏幕上占据的字符单元树和在内存中的字节数是一样的。 在 UTF-8 的 locale 环境下，字符串就被折散了。下面的patch可以在多字节的locale环境下将它们转回到英文信息：

```
patch -Np1 -i ../texinfo-4.8-multibyte-1.patch
```

Texinfo 允许本地用户通过位于临时文件中的符号连接改写任意文件，下面的补丁可以修正这个问题：

```
patch -Np1 -i ../texinfo-4.8-tempfile_fix-2.patch
```

为编译 Texinfo 做准备：

```
./configure --prefix=/usr
```

编译软件包：

```
make
```

要测试结果，请运行： **`make check`** 。

安装软件包：

```
make install
```

安装 texinfo 组件中本应由 TeX 来安装的部份(可选操作)：

```
make TEXMF=/usr/share/texmf install-tex
```

make 参数的含义：

`TEXMF=/usr/share/texmf`

如果你以后打算安装 TeX 的话，makefile 中的 `TEXMF` 变量保存着你的 TeX 树的位置。

Info 文档系统使用一个纯文本文件来存放菜单条目的列表。这个文件位于 `/usr/share/info/dir` 。不幸的是，由于不同软件包中 Makefile 的偶然问题，有时候这个文件会与实际安装在系统里的 Info 文档不一致。如果你要再次创建 `/usr/share/info/dir` 文件，可以使用下面的命令：

```
cd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.54.2. Texinfo 的内容

安装的程序： info, infokey, install-info, makeinfo, texi2dvi, texi2pdf, texindex

简要描述

`info`	用于读取 Info 文档，它类似于 man 手册页，但是他们会给出更深入的解释而不是停留在解释程序参数上。例如你可以看看 `man bison` 和 `info bison` 的区别。
`infokey`	把包括 Info 设置的源文件编译成二进制格式
`install-info`	安装 info 文档，它会更新 `info` 的索引文件
`makeinfo`	将 Texinfo 源文档翻译成不同的格式，包括html、info文档、文本文档。
`texi2dvi`	把给定的 Texinfo 文档格式化成可打印的设备无关的文件
`texi2pdf`	将 Texinfo 文档转化成 PDF 文件
`texindex`	对 Texinfo 索引文件进行排序

6.55. Udev-096

Udev 软件包包含动态地创建设备节点的程序。

预计编译时间： 0.1 SBU所需磁盘空间： 6.8 MB

6.55.1. 安装 Udev

udev-config 压缩包里面包含用配置 Udev 的 LFS-specific 文件。把它解压到 Udev 的源码目录：

```
tar xf ../udev-config-6.2.tar.bz2
```

创建一些Udev无法创建的设备和目录，因为这些会在系统启动的早些时候被用到：

```
install -dv /lib/{firmware,udev/devices/{pts,shm}}
mknod -m0666 /lib/udev/devices/null c 1 3
ln -sv /proc/self/fd /lib/udev/devices/fd
ln -sv /proc/self/fd/0 /lib/udev/devices/stdin
ln -sv /proc/self/fd/1 /lib/udev/devices/stdout
ln -sv /proc/self/fd/2 /lib/udev/devices/stderr
ln -sv /proc/kcore /lib/udev/devices/core
```

编译软件包：

```
make EXTRAS="extras/ata_id extras/cdrom_id extras/edd_id \
            extras/firmware extras/floppy extras/path_id \
            extras/scsi_id extras/usb_id extras/volume_id"
```

make 选项的含义：

`EXTRAS=...`

这将会编译一些帮助程序，对定制Udev的规则很有帮助。

要测试结果，请运行： **`make test`** 。

注意，Udev的测试单元会在宿主系统的日志中产生很多信息。这些都是无害的，可以被忽略掉。

安装软件包：

```
make DESTDIR=/ \
    EXTRAS="extras/ata_id extras/cdrom_id extras/edd_id \
            extras/firmware extras/floppy extras/path_id \
            extras/scsi_id extras/usb_id extras/volume_id" install
```

make 参数的含义：

`DESTDIR=/`

防止编译 Udev 的进程杀死可能存在于宿主系统中的 `udevd` 进程。

Udev 要工作，需要配置才可以。因为默认是不安装任何配置文件的。安装 LFS-specific 配置文件：

```
cp -v udev-config-6.2/[0-9]* /etc/udev/rules.d/
```

安装解释如何创建 Udev 规则的文档：

```
install -m644 -D -v docs/writing_udev_rules/index.html \
    /usr/share/doc/udev-096/index.html
```

6.55.2. Udev 的内容

安装的程序： ata_id, cdrom_id, create_floppy_devices, edd_id, firmware_helper, path_id, scsi_id, udevcontrol, udevd, udevinfo, udevmonitor, udevsettle, udevtest, udevtrigger, usb_id, vol_id, write_cd_aliases安装的目录： /etc/udev

简要描述

`ata_id`	为 Udev 提供关于 ATA 驱动器的一个唯一的字符串和一些附加信息（uuid，label等）
`cdrom_id`	为 Udev 提供 CD-ROM 或 DVD-ROM 驱动器的性能
`create_floppy_devices`	创建所有可能的 CMOS 类型的 floppy 设备
`edd_id`	为 Udev 提供关于 BIOS 磁盘驱动器的 EDD ID
`firmware_helper`	为设备加载 firmware
`path_id`	提供设备的最短的唯一的硬件路径
`scsi_id`	根据向特定设备发送SCSI INQUIRY命令的返回信息，为 Udev 提供一个唯一的 SCSI 标识符
`udevcontrol`	为运行 `udevd` 守护进程，配置一些选项。比如，log level。
`udevd`	一个守护进程，侦听热插拔事件，并针对事件，创建设备，运行配置好的外部程序。
`udevinfo`	允许用户查询 `udev` 数据库以得到当前这个系统上所有设备的信息，它也提供一种方式去查询任何设备在 `sysfs` 树里去帮助创建 Udev 规则。
`udevmonitor`	打印出从Udev的规则运行之后，收到的内核事件和 Udev 发出的环境变量。
`udevsettle`	监视 Udev 的事件队列，如果当前热插拔事件被处理完就立即退出。
`udevtest`	模拟一个 `udev` 为那些给定的设备,并且打印出真实节点的名称 `udev` 可能已经被创建或者(不在LFS中)被重命名的网络接口。
`udevtrigger`	重新切换到内核空间的热插拔事件处理
`usb_id`	为 Udev 提供关于 USB 设备的信息
`vol_id`	为 Udev 提供一个文件系统的 label 和 uuid
`/etc/udev`	包含 `udev` 配置文件、设备许可、设备命名规则。

6.56. Util-linux-2.12r

Util-linux 软件包包含许多工具。其中比较重要的是加载、卸载、格式化、分区和管理硬盘驱动器，打开 tty 端口和得到内核消息。

预计编译时间： 0.2 SBU所需磁盘空间： 17.2 MB

6.56.1. FHS 兼容性说明

FHS 推荐使用 `/var/lib/hwclock` 目录代替常用的 `/etc` 目录以定位 `adjtime` 文件。要将 `hwclock` 编译成与 FHS 兼容的程序，运行下面的命令：

```
sed -i 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
    hwclock/hwclock.c
mkdir -p /var/lib/hwclock
```

6.56.2. 安装 Util-linux

Util-linux 在基于新版本的 Linux-Libc-Headers 编译时会出错，下面的补丁修正了这个问题：

```
patch -Np1 -i ../util-linux-2.12r-cramfs-1.patch
```

为编译 Util-linux 做准备：

```
./configure
```

编译软件包：

```
make HAVE_KILL=yes HAVE_SLN=yes
```

make 参数的含义：

`HAVE_KILL=yes`

防止编译和安装 `kill` 程序(已经由 Procps 安装了)。

`HAVE_SLN=yes`

防止编译 `sln` 程序(这是静态连接的 `ln` ，已经由 Glibc 安装了)。

这个软件包没有附带测试程序。

安装软件包：

```
make HAVE_KILL=yes HAVE_SLN=yes install
```

6.56.3. Util-linux 的内容

安装的程序： agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, flock, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, pg, pivot_root, ramsize(→rdev), raw, rdev, readprofile, rename, renice, rev, rootflags(→rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff(→swapon), swapon, tailf, tunelp, ul, umount, vidmode(→rdev), whereis, write

简要描述

`agetty`	打开 tty 端口，为登录名称建立命令控制符，并引出 `login` 程序
`arch`	报告机器的体系结构
`blockdev`	在命令行中调用块设备的 ioctl
`cal`	显示一个简单的日历
`cfdisk`	处理指定设备的分区表
`chkdupexe`	找出重复的可执行文件
`col`	过滤回显反馈线
`colcrt`	过滤那些 `nroff` 终端不具备输出的能力，比如高分点距、半线距
`colrm`	过滤掉给出的列
`column`	把输出格式化为几列
`ctrlaltdel`	设置 CTRL+ALT+DEL 组合键的功能为硬重启或软重启
`cytune`	查询和修改 Cyclade 驱动器的中断入口
`ddate`	把阳历日期转换为 Discordian 日期
`dmesg`	显示内核的启动信息
`elvtune`	调整块设备的相互作用和性能
`fdformat`	低级格式化一张软盘
`fdisk`	磁盘分区管理程序
`flock`	得到一个文件锁，并根据状态执行一个命令

`fsck.cramfs`	对 Cramfs 文件系统的一致性进行检查
`fsck.minix`	对 Minix 文件系统的一致性进行检查
`getopt`	在给出的命令行进行选项和参数解析
`hexdump`	用用户指定的方式(包括ASCII, 十进制, 十六进制, 八进制)显示一个文件或者标准输入的数据
`hwclock`	查询和设置硬件时钟(也被称为 RTC 或 BIOS 时钟)
`ipcrm`	删除给定的进程间通信(IPC)资源
`ipcs`	提供 IPC 状态信息
`isosize`	报告 iso9660 文件系统的大小
`line`	单行拷贝
`logger`	设置系统日志的入口
`look`	显示以某个给定字符串开头的行
`losetup`	启动和控制回环(loop)设备
`mcookie`	为 `xauth` 生成 magic cookies (128位的随机16进制数)
`mkfs`	在一个设备(通常是一个硬盘分区)设备上建立文件系统
`mkfs.bfs`	创建一个 Santa Cruz Operations (SCO) bfs 文件系统
`mkfs.cramfs`	创建 cramfs 文件系统
`mkfs.minix`	创建 Minix 文件系统
`mkswap`	初始化指定设备或文件，以用做交换分区
`more`	分屏显示文件，但没有 less 好用
`mount`	把一个文件系统从一个设备挂载到一个目录
`namei`	显示指定路径的符号链接
`pg`	显示文本文件内容，一次显示一屏
`pivot_root`	使某个文件系统成为当前进程的根文件系统
`ramsize`	显示或者改变 RAM disk 的大小
`raw`	将一个原始的 Linux 字符设备绑定到一个块设备
`rdev`	查询和设置内核的根设备和其他信息
`readprofile`	显示内核侧写文件 /proc/profile 的信息
`rename`	对文件进行重命名
`renice`	修改正在运行进程的优先级
`rev`	颠倒一个文件每行字符的顺序
`rootflags`	在挂载根设备时查询和设置额外的信息

script	为终端会话过程建立一个 typescipt 文件，记录会话过程中终端的输出。
setfdprm	设置用户定义的软盘参数
setsid	在一个新的会话中运行程序
setterm	设置终端属性
sfdisk	磁盘分区表管理工具
swapoff	取消对指定交换设备和交换文件的使用
swapon	使指定的交换设备和交换文件生效
tailf	跟踪一个日志文件，显示日志的最后10行，并将日志中新的记录也显示出来。
tunelp	设置打印设备的参数
ul	用来将指定文件中出现的下划线使用指定终端画下横线的序列
umount	卸载一个被挂载的文件系统
vidmode	查询和设置视频模式
whereis	确定某命令二进制文件、源文件、手册文档的位置
write	发一个消息给另一个用户，如果他开启了 writting 的话。

6.57. Vim-7.0

Vim 软件包包含一个强大的文本编辑器。

预计编译时间： 0.4 SBU所需磁盘空间： 47.4 MB

Alternatives to Vim

如果你更喜欢使用其他的编辑器，比如Emacs, Joe, Nano，请参考这里：*http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html* 获取安装方法。

6.57.1. 安装 Vim

首先，将 `vim-7.0.tar.bz2` 和 `vim-7.0-lang.tar.gz` (可选) 解包到同一个目录下。然后应用几个patch来修正一些 Vim-7.0 的bug：

```
patch -Np1 -i ../vim-7.0-fixes-7.patch
```

这个版本的 Vim 会安装翻译过的 man 手册，并把它们放到 Man-DB 搜索不到的目录下。给 vim 打补丁来使其安装到可搜索目录里面，并允许Man-DB在运行的时候转换成希望的格式：

```
patch -Np1 -i ../vim-7.0-mandir-1.patch
```

因为上面的patch中的一个，导致通过Http来下载拼写文件时候会出现问题。开发者解决了这个问题，应用下面的patch就可以解决：

```
patch -Np1 -i ../vim-7.0-spellfile-1.patch
```

最后把配置文件 `vimrc` 的默认位置修改到 `/etc` 目录中：

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

为编译 Vim 做准备：

```
./configure --prefix=/usr --enable-multibyte
```

配置选项的含义：

`--enable-multibyte`

我们强烈推荐你启用该选项(虽然它是可选的)，因为它使得 Vim 可以支持使用多字节字符编码的文件，在一个使用多字节字符集的 locale 上，这是必需的。另外该选项还有助于编辑在默认使用 UTF-8 字符集的其它 Linux 发行版(如 Fedora Core)上创建的文本文件。

编译软件包：

```
make
```

要测试结果，请运行：`make test` 。注意，Vim 的测试套件会输出一大堆混乱的二进制字符到屏幕上，有时会把当前的终端设置搞乱，所以可以考虑将其输出重定向到一个日志文件。

安装软件包：

```
make install
```

在 UTF-8 的 locale 下，`vimtutor` 程序会将教程从 ISO-8859-1 转化为 UTF-8。如果一些教程不是 ISO-8859-1 编码的，将会导致不可读。如果你解压了 `vim-7.0-lang.tar.gz` 包，并打算使用 UTF-8 的 locale，删除非ISO-8859-1编码的教程。英文版本的教程将会被安装：

```
rm -f /usr/share/vim/vim70/tutor/tutor.{gr,pl,ru,sk}
rm -f /usr/share/vim/vim70/tutor/tutor.??.*
```

许多用户习惯于使用 `vi` 而不是 `vim` ，下面的命令为适应这种习惯创建一个二进制文件以及man文档的符号链接：

```
ln -sv vim /usr/bin/vi
for L in "" fr it pl ru; do
    ln -sv vim.1 /usr/share/man/$L/man1/vi.1
done
```

默认情况下，Vim的文档被安装在 `/usr/share/vim` 目录下。下面的符号链接允许通过 `/usr/share/doc/vim-7.0` 来访问，使其与其他软件包的文档保持一致：

```
ln -sv ../vim/vim70/doc /usr/share/doc/vim-7.0
```

如果您计划在您的 LFS 系统上安装 X Window 系统，那么您可能必须在安装 X 之后重新编译 Vim 。Vim 有带有图形用户接口的版本，但需要安装 X 和其它一些库。要了解更多关于这方面的信息请参考 Vim 的安装手册和 BLFS 中有关 Vim 安装指导的页面：*http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.57.2. 配置 Vim

在默认情况下，`vim` 是以与 vi 兼容的模式运行，这可能对使用过其它文本编辑器的人来说感到陌生。有些人可能喜欢这种模式，但是我们强烈建议使用 vim 模式运行 vim 。下面的配置文件明确的设置了"nocompatible"模式(也可以改用"compatible"模式)，这是必要的，因为如果要改变或覆盖其它设置，必须要在这个设置之后。使用下面的命令创建一个默认的 `vim` 配置文件：

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term =="iterm") || (&term =="putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

`set nocompatible` 将使 `vim` 以比默认的 vi 兼容模式功能更强的方式运行。你可以去掉"no"来保持默认的 `vi` 模式。*`set backspace=2`* 让退格键能跨行、自动缩进、插入。*`syntax on`* 打开了 vim 的语法高亮功能。最后，带有 *`set background=dark`* 的 *if* 语句纠正了 `vim` 对于终端背景颜色的猜测。对于那些使用黑色背景的用户这是一个较好的配色方案。

使用下面的命令可以得到其它可以使用的选项的说明文档：

```
vim -c ':options'
```

注意

默认情况下，Vim仅会安装英文的拼写检查文件。要安装你熟悉的语言的拼写检查文件，从 *ftp://ftp.vim.org/pub/vim/runtime/spell/* 下载与你的语言相关的 `*.spl` 文件和 `*.sug` 文件（可选），以及字符解码器。把它们保存到 `/usr/share/vim/vim70/spell/` 目录下。

要使用这些拼写检查文件，在 `/etc/vimrc` 里面需要配置一些选项，例如：

```
set spelllang=en,ru
set spell
```

要获取更多的信息，可以查看上面 URL 上的 README 文件。

6.57.3. Vim 的内容

安装的程序： efm_filter.pl, efm_perl.pl, ex(→vim), less.sh, mve.awk, pltags.pl, ref, rview(→vim), rvim(→vim), shtags.pl, tcltags, vi(→vim), view(→vim), vim, vim132, vim2html.pl, vimdiff(→vim), vimm, vimspell.sh, vimtutor, xxd

简要描述

`efm_filter.pl`	能读取 `vim` 产生的错误的一个过滤器
`efm_perl.pl`	将 perl 解释器的错误信息重新格式化以用于 `vim` 的"quickfix"模式。
`ex`	以 ex 模式启动 `vim`
`less.sh`	用 less.vim 起动 `vim` 的脚本
`mve.awk`	处理 `vim` 错误信息
`pltags.pl`	为 Perl 代码建立一个标记文件，以用于 `vim`
`ref`	检查参数的拼写错误
`rview`	一个 `view` 的限制版，它不能启动任何 shell 命令并且 `view` 不能被挂起。
`rvim`	一个 `vim` 的限制版。它不能启动任何 shell 命令并且 `vim` 不能被挂起。
`shtags.pl`	为 perl 代码产生一个标志文件
`tcltags`	为 TCL 代码产生一个标志文件
`view`	以只读模式启动 `vim`
`vi`	`vim` 的一个链接
`vim`	编辑器
`vim132`	在132列模式下的终端中起动 `vim`
`vim2html.pl`	将 vim 文档转化为 HTML 格式
`vimdiff`	使用 `vim` 同时编辑一个文档的 2 或 3 个版本并显示他们的区别使用 `vim` 同时编辑一个文档的 2 或 3 个版本并显示他们的区别
`vimm`	在一个远端终端上开启 DEC 定位输入模式
`vimspell.sh`	检查文件拼写并产生一个需要在 `vim` 中强调的语法报告。这个脚本需要 LFS 和 BLFS 都不提供的旧的 Unix `spell` 命令
`vimtutor`	教你使用 `vim` 的基本操作和命令
`xxd`	生成十六进制转储或者做相反的工作，因此它能给二进制文件打补丁。

6.58. 关于调试符号

在缺省情况下，大多数程序和库都是带调试符号(使用 `gcc` 的 `-g` 选项)编译的。 当调试一个带调试符号的程序时，调试器不仅能给出内存地址，还能给出函数和变量的名字。

但是，这些调试符号明显地增大了程序和库。想知道这些调试符能带来多大的差异，请看下面的统计资料：

- 带调试符号的动态 `bash` 二进制文件：1200 KB
- 不带调试符号的动态 `bash` 二进制文件：480 KB
- 带调试符号的 Glibc 和 GCC 文件 (位于 `/lib` 和 `/usr/lib` 目录)：87 MB
- 不带调试符号的 Glibc 和 GCC文件：16 MB

根据使用的编译器和连接动态程序的 C 库的版本的不同，文件的大小可能会有所不同，但是比较带调试符号与不带调试符号的程序的比较结果应该不会改变，大概是 2~5 倍大小。

由于大多数人都不会在系统软件上使用调试器，把这些符号去掉就能节省大量的空间。下一节将给您展示如何从程序和库文件中去除所有调试符号链接。附加的信息在系统优化信息里可以找到 *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*。

6.59. 再次清理系统

如果编译此系统的用户不是一个程序员并且不打算在系统软件上做任何调试工作，系统大小可以通过从二进制包和库文件中删除调试链接削减大约 90 MB 的空间，这样做的代价是您以后将不能随时调试系统。

大多数人可以毫无困难地凭经验使用命令提示。但是，很容易因为一个打字错误就被报告新系统不能使用，因此在运行 `strip` 命令前，对系统当前数据做一个备份是一个不错的主意。

在执行清理命令之前，请特别注意确保正在运行的二进制文件不被清理。如果您不确定是否用户是进入在虚根环境（节 6.4, "进入 Chroot 环境"）下操作，请首先退出虚根环境：

```
logout
```

接着用下面的命令再次进入：

```
chroot $LFS /tools/bin/env -i \
    HOME=/root TERM=$TERM PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /tools/bin/bash --login
```

现在二进制文件和库文件能安全地被清理了：

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
  -exec /tools/bin/strip --strip-debug '{}' ';'
```

很多文件将被报告说不能识别它们的文件格式，这些警告可以安全地忽略。这些警告只是表明那些文件是脚本而不是二进制文件。

如果硬盘空间非常紧张，这个 *`--strip-all`* 选项可以被用在二进制文件目录 `/{,usr/}{bin,sbin}` 中以获得更多的空间。不要在库文件里使用这个参数，因为这个参数将会破坏库文件。

6.60. 最终的清理

从现在起，在退出后，每当重新进入 Chroot 环境，请使用下面修改过的 chroot 命令:

```
chroot "$LFS" /usr/bin/env -i \
    HOME=/root TERM="$TERM" PS1='\u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /bin/bash --login
```

这样做的理由是起初在 `/tools` 目录下的程序已经不再需要了，这个目录可以删除以重新获得更多硬盘空间。在实际动手删除这个目录之前，请退出虚根环境并且使用上面修改过的命令重新进入虚根环境。还有一点，在删除 `/tools` 目录前，请将此目录打包压缩并且存储在一个安全的地方，以备以后编译另一个 LFS 系统。

注意

删除 `/tools` 目录将也会删除临时文件夹中已拷贝的 Tcl, Expect, DejaGNU 等文件,而这些文件是那些正在运行的工具链接测试程序使用的。如果要在以后使用这些程序，它们需要重新编译和重新安装，BLFS 手册介绍了如何重新安装(参见 *http://www.linuxfromscratch.org/blfs/*)。

如果虚拟内核文件系统被卸载了，或者是重启系统了。确保虚拟内核文件系统在重新进入 chroot 环境时已经挂载了（参见 节 6.2.2, "挂载并填充 /dev 目录" 和 节 6.2.3, "挂载虚拟内核文件系统"）。

7. 配置系统启动脚本

7.1. 简介

本章详细描述了如何安装并恰当配置启动脚本(LFS-Bootscripts)，大多数脚本无需修改就可运行，但少数需要附加配置文件，因为它们与硬件相关。

因为 System-V 风格的初始化脚本被广泛使用，因此本书使用这种风格的脚本，作为补充选项，在 *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt* 可以找到一个详细设计 BSD 风格启动初始化的提示。在 LFS 邮件列表里搜索"depinit"还可以找到更多选择。

如果您使用其它风格的初始化脚本，请跳过本章继续进行 Chapter 8 。

7.2. LFS-Bootscripts-6.2

LFS-Bootscripts 软件包包含一套在 LFS 系统启动和关闭时的启动和停止脚本。

预计编译时间： 0.1 SBU所需磁盘空间： 0.3 MB安装依赖于： Bash, Coreutils

7.2.1. 安装 LFS-Bootscripts

安装软件包：

```
make install
```

7.2.2. Contents of LFS-Bootscripts

Installed scripts: checkfs, cleanfs, console, functions, halt, hotplug, ifdown, ifup, localnet, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysklogd, template, udev

简要描述

`checkfs`	在挂载之前检查文件系统完整性(日志文件系统和基于网络的文件系统除外)
`cleanfs`	删除系统重启后就不需要保存了的文件，例如在 `/var/run/` 和 `/var/lock/` 目录下的文件；重新创建 `/var/run/utmp` 文件并删除可能存在的 `/etc/nologin` ， `/fastboot` ， `/forcefsck` 文件。
`console`	为指定的键盘布局读入正确的键盘映射表，并设置屏幕字体。
`functions`	包含在不同脚本中共用的一些函数，例如错误和状态检查函数。
`halt`	关闭系统
`hotplug`	为系统设备加载模块
`ifdown`	协助 network 脚本停止网络设备
`ifup`	协助 network 脚本启动网络设备
`localnet`	设置系统主机名和本地回环(loopback)设备
`mountfs`	挂载所有文件系统，有 *noauto* 标记或者基于网络的文件系统除外。
`mountkernfs`	用来挂载内核提供的文件系统，例如 `proc`
`network`	设置网络连接，例如网卡等；并设置默认网关(如果可用) 。
`rc`	主要的运行级控制脚本，负责让所有其它脚本按符号链接名确定的顺序一个接一个的运行。
`reboot`	重新启动系统
`sendsignals`	在系统重启或关闭系统之前，确保每一个进程都已经终止了。
`setclock`	如果硬件时钟没有设置为 UTC 时间，将内核时钟重置为本地时间。
`static`	提供为网络接口指派静态 IP 地址的功能
`swap`	启用或禁用交换文件和交换分区
`sysklogd`	启动或停止系统和内核日志守护进程
`template`	为其它守护进程创建自定义启动脚本的模板
`udev`	启动 udev 并在 `/dev` 目录创建设备节点

7.3. 启动脚本是如何工作的?

Linux 使用的是基于 运行级*(run-levels)* 概念的称为 SysVinit 的专用启动工具。它在不同的系统上可能是完全不一样的，所以不能认为一个脚本在某个 Linux 发行版上工作正常，于是在 LFS 中也会正常工作。LFS 有自己的一套规则，当然，LFS 也遵守一些公认的标准。

SysVinit(从现在开始我们称之为"init")以运行级的模式来工作，一般有 7 个运行级(从 0 到 6，实际上可以有更多的运行级，但都是用于特殊情况而且一般使用不到。 参见 `init(8)` 以获得更多信息)，每个运行级对应于一套设定好的任务，当启动一个运行级的时候，计算机就需要执行相应的任务。默认的运行级是 3，下面是对不同运行级的描述：

0: 停止计算机 1: 单用户模式 2: 无网络多用户模式 3: 有网络多用户模式 4: 保留作自定义，否则同运行级 3 5: 同运行级 4，一般用于图形界面(GUI)登录(如 X 的 `xdm` 或者 KDE 的 `kdm`) 6: 重新启动计算机

用来改变运行级的命令是 `init _` [runlevel] `_` ，这里的 `[runlevel]` 是目标运行级。例如，要重启计算机，用户可以运行 `init 6` 命令， `reboot` 其实只是这个命令的别名，同样， `halt` 命令也只是 `init 0` 的别名。

在 `/etc/rc.d` 目录下有一些类似于 `rc?.d` 的目录(这里 ? 是运行级的数字)以及 `rcsysinit.d` ，里面都包含许多符号链接，其中一些以 *K* 字母开头，另外一些以 *S* 字母开头，这些链接名在首字母后面都跟着两个数字。K 字母的含义是停止(杀死)一个服务，S 字母的含义是启动一个服务。而数字则确定这些脚本的启动顺序，从 00 到 99(数字越小执行的越早)。当 `init` 转换到其它运行级时，一些相应的服务会停止，而另一些服务则会启动。

真正的脚本则在 `/etc/rc.d/init.d` 目录下，它们完成实际工作，符号链接都是指向它们的。停止脚本的链接和启动脚本的链接都指向 `/etc/rc.d/init.d` 目录下同一个脚本，这是因为调用这些脚本时可以使用不同的参数，例如 `start` , `stop` , `restart` , `reload` , `status` 当调用 K 链接时，相应的脚本用 `stop` 参数运行；当调用 S 链接时，相应的脚本用 `start` 参数运行。

上面的说明有一个例外，在 `rc0.d` 和 `rc6.d` 目录下以 *S* 开头的链接不会启动任何东西，而是用 `stop` 参数调用，来停止某些服务。这背后的逻辑是，当用户要重启或关闭系统的时候，不会要启动什么服务，只会要系统停止。

以下是脚本参数的描述：

`start`

启动服务

`stop`

停止服务

`restart`

停止服务，然后再启动

`reload`

该服务的配置已更新。如果修改了某个服务的配置文件，又不必重启这个服务的时候，可以使用这个参数。

`status`

显示服务的状态，如果服务正在运行，会显示该服务进程的 PID 。

您可以自由修改启动进程工作的方式(毕竟这是您自己的 LFS 系统)，我们这里给出的文件只是它们怎样工作的一个示例而已。

7.4. LFS 系统的设备和模块处理

在 Chapter 6 里，我们安装了 Udev 软件包，在开始深入讨论它如何工作之前，我们先简要回顾一下以前处理设备的方法。

传统上一般 Linux 系统使用创建静态设备的方法，因此在 `/dev` 目录下创建了大量的设备节点(有时会有数千个节点)，而不管对应的硬件设备实际上是否存在。这通常是由 `MAKEDEV` 脚本完成的，这个脚本包含许多调用 `mknod` 程序的命令，为这个世界上可能存在的每个设备创建相应的主设备号和次设备号。而使用 udev 方式的时候，只有被内核检测到的设备才为其创建设备节点。因为每次系统启动的时候都要重新创建这些设备节点，所以它们被存储在 `tmpfs` 文件系统(一种完全存在于内存里，不占用任何磁盘空间的文件系统)上，设备节点不需要很多磁盘空间，所占用的内存可以忽略不计。

7.4.1. 历史

2000 年 2 月的时候，2.3.46 版本的内核引入了一种称为 `devfs` 的文件系统，在 2.4 系列稳定版本的内核中都是可用的。尽管它存在于内核源代码中，但这种动态创建设备的方法却从未得到核心内核开发者们的全力支持。

`devfs` 存在的主要的问题是它处理设备检测、创建和命名的方式，其中设备节点的命名可能是最严重的问题。一般可接受的方式是，如果设备名是可配置的，那么设备命名 策略应该由系统管理员决定，而不是由某些开发者强制规定。 `devfs` 文件系统还存在竞争条件(race conditions)的问题，这是它天生的设计缺陷，不对内核做彻底的修改就无法修正这个问题。因为近来缺乏维护，它已经被标记为 deprecated(反对的)。

随着非稳定的 2.5 内核树的开发，即后来发布的 2.6 系列稳定版本内核，一种被称为 `sysfs` 的新虚拟文件系统诞生了。 `sysfs` 的工作是把系统的硬件配置视图导出给用户空间的进程。由于有了这个用户空间可见的表示，代替 `devfs` 方案的时机就成熟了。

7.4.2. Udev 实现

7.4.2.1. Sysfs 文件系统

上面简单的提到了 `sysfs` 文件系统，您可能想知道 `sysfs` 是怎么认出系统中存在的设备以及应该使用什么设备号。对于已经编入内核的驱动程序，当被内核检测到的时候，会直接在 `sysfs` 中注册其对象；对于编译成模块的驱动程序，当模块载入的时候才会这样做。一旦挂载了 `sysfs` 文件系统(挂载到 `/sys`)， 内建的驱动程序在 `sysfs` 注册的数据就可以被用户空间的进程使用，并提供给 `udev` 以创建设备节点。

7.4.2.2. Udev 启动脚本

`S10udev` 初始化脚本负责在 Linux 启动的时候创建设备节点，该脚本首先将 `/sbin/udevsend` 注册为热插拔事件处理程序。热插拔事件(随后将讨论)本不应该在这个阶段发生，注册 `udev` 只是为了以防万一。然后 `udevstart` 遍历 `/sys` 文件系统，并在 `/dev` 目录下创建符合描述的设备。例如， `/sys/class/tty/vcs/dev` 里含有"7:0"字符串， `udevstart` 就根据这个字符串创建主设备号为 *7* 、次设备号为 *0* 的 `/dev/vcs` 设备。 `udevstart` 创建的每个设备的名字和权限由 `/etc/udev/rules.d/` 目录下的文件指定的规则来设置，这些文件以类似于 LFS 启动脚本风格的编号。如果 `udev` 找不到所创建设备的权限文件，就将其权限设置为缺省的 *660* ，所有者为 *root:root* 。

上面的步骤完成后，那些已经存在并且已经内建驱动的设备就可以使用了，那么以模块驱动的设备呢？

前面我们提到了"热插拔事件处理程序"的概念，当内核检测到一个新设备连接时，内核会产生一个热插拔事件，并在 `/proc/sys/kernel/hotplug` 文件里查找处理设备连接的用户空间程序。 `udev` 初始化脚本将 `udevsend` 注册为该处理程序。当产生热插拔事件的时候，内核让 `udev` 在 `/sys` 文件系统里检测与新设备的有关信息，并为新设备在 `/dev` 里创建项目。

这样带来了 `udev` 存在的一个问题，之前 `devfs` 也存在同样的问题。这通常是个"先有鸡 还是先有蛋"问题。大多数 Linux 发行版通过 `/etc/modules.conf` 配置文件来处理模块加载，对某个设备节点的访问导致相应的内核模块被加载。对 `udev` 这个方法就行不通了，因为在模块加载前，设备节点根本不存在。为了解决这个问题，在 LFS-Bootscripts 软件包里加入了 `S05modules` 启动脚本，以及 `/etc/sysconfig/modules` 文件。通过在 `modules` 文件里添加模块名，就可以在系统启动的时候加载这些模块，这样 `udev` 就可以检测到设备，并创建相应的设备节点了。

注意，在慢速的机器上，或者对于需要创建大量设备节点的驱动程序，创建设备的过程可能需要好几秒钟，这意味着某些设备节点不能立即访问到。

7.4.2.3. 设备节点创建

为了得到正确的主次设备号，Udev 依赖于 `/sys 目录下的` `sysfs 提供的信息。` 例 如， `/sys/class/tty/vcs/dev` 包含字符串 "7:0"，被 `udevd` 用来创建主设备号是7、次设备号是0的设备节点。在 `/dev` 目录下创建的设备节点的名字和权限由 `/etc/udev/rules.d/ 目录下相应的规则决定。` 这 些以类似的形式包含在 LFS-Bootscripts 包中。如果 `udevd` 不能为它现在创建的设备发现一个规则，它会默认把权限设置为 660 ，属主设置为 _root: root。_Udev 规则语法的文档可以查看 `/usr/share/doc/udev -096/index.html。`

7.4.2.4. 模块加载

被编译成模块的设备驱动可能有编译进去的别名。别名可以通过 `modinfo` 程序的输出来查看，通常与设备总线特 有的标识有关（模块要支持）。例如，*snd-fm801* 驱动支持 PCI 设备，通过生产厂商 ID 0x1319 和设备 ID 0x0801，有一个别名"pci:v00001319d00000801sv*sd*bc04sc01i*"。 对于大多数的设备，总线驱动通过 `sysfs`

导出可能会处理的设备驱动的别名。例如，`/sys/bus/pci/devices/0000:00:0d.0/modalias` 文件可能包含字符串"pci:v00001319d00000801sv00001319sd00001319bc04sc01i00"。 LFS 安装的规则会导致 `udevd` 调用 `/sbin/modprobe` 处理热插拔事件环境变量 `MODALIAS` 的内容（应该与 sysfs 中的 modalias 文件的内容一样）。 因此在通配符扩展之后，加载所有的别名匹配这个字符串的模块。

这个例子中，除了 *snd-fm801*，荒废的（不想要的） *forte* 驱动会被加载（如果它是有效的）。 查看下面的方法来避免加载不想要的模块。

内核能在后台加载有关网络协议、文件系统和 NLS（国际语言支持） 支持的模块。

7.4.2.5. 处理可热插拔/动态设备

当您插入一个设备，例如一个 USB 接口的 MP3 播放器，内核会检测到设备连接，并产生一个热插拔事件，如果驱动程序已经加载(要么是因为驱动已经编入内核，要么是已经通过 `S05modules` 启动脚本加载了)，`udev` 将被调用，并根据 `/sys` 目录下的 `sysfs` 数据来创建相应的设备节点。如果该设备的驱动是一个未加载的模块，将设备连接到系统上只会让内核的总线驱动产生一个热插拔事件，通知用户空间有新设备连 接，但并不加载驱动。事实上，什么都没有做，设备仍然不能使用。

如果刚才插入的设备有一个驱动程序模块但是尚未加载，Hotplug 软件包就有用了，它就会响应上述的内核总线驱动热插拔事件并加载相应的模块，为其创建设备节点，这样设备就可以使用了。

7.4.3. 创建设备的问题

自动创建设备节点的时候，存在一些已知的问题：

7.4.3.1. 内核模块没有自动加载

如果有一个总线特有的别名，并且总线驱动器导出必需的别名到 `sysfs，` 那么 Udev 将只会加载一个模块。在其他情况下，需要安排其他的模块加载方式。在 Linux-2.6.16.27 中，Udev 为INPUT, IDE, PCI, USB, SCSI, SERIO 和 FireWire 设备加载适当的驱动。

为了确定你请求的设备驱动 Udev是否支持，可以以模块的名字为参数运行 `modinfo`。 现在把设置设备目录到 `/sys/bus` ，检测那里是否有一个 `modalias` 文件。

如果 `modalias` 文件存在于 `sysfs` 中，驱动程序支持设备可以直接通信，但是没有别名，这是驱动中的一个 bug。不利用 Udev 的帮助加载驱动，希望在以后这个问题会被修正。

如果在 `/sys/bus下的` 相应目录内没有 `modalias` 文件，这就意味着内核开发者没有添加对这种总线类型的模块别名支持。在 Linux-2.6.16.27中，ISA 总线就是这种情况。希望在下个内核版本中会修正这个问题。

Udev 根本不会加载 "封装" 驱动像 *snd-pcm-oss*，和 非硬件驱动器像 *loop*。

7.4.3.2. 内核模块没有自动加载，并且 Udev 也不加载

如果 “封装” 模块增强其他模块提供的功能（例如，*snd-pcm-oss* 增强 *snd-pcm* 的功能，使声卡对于 OSS 程序可用），配置 `modprobe` 使其在加载了相应模块后加载其封装模块。在 `/etc/modprobe.conf 中` 添加一个 "install" 行来实现，例如：

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

如果模块不是一个封装，而是对自己本身有用，配置 `S05modules` 启动脚本使其在系统启动的时候加载。 要实现这个，在 `/etc/sysconfig/modules 文件的相应行上添加模块的名字。` 这对于封装模块也有用，但不是最优的。

7.4.3.3. Udev 加载了不需要的模块

在下面例子中，对于 *forte* 模块，可以不编译它或者是在 `/etc/modprobe.conf 文件中` 列入黑名单：

```
blacklist forte
```

列入黑名单的模块仍然能够通过 `modprobe` 命令手工加载。

7.4.3.4. Udev 错误的创建了设备或创建了错误的符号链接

如果规则匹配不是预想的设备，那么这种情况就会经常发生。例如，一个写的很糟糕的规则通过计算机提供商匹配到一个 SCSI 硬盘（期望的）和一个一般的 SCSI 设备（错误的）。找出这些不合格的规则，更正他们。

7.4.3.5. Udev 规则工作不可靠

这或许是前面问题的一个表现。如果不是，你的规则使用 `sysfs` 属性，这可能是一个内核计时问题，在下个版本的内核中会被修正 。 现在，你可以通过创建一个等待使用 `sysfs 属性` 的规则，把它添加到 `/etc/udev/rules.d/10-wait_for_sysfs.rules` 文件中。如果你做了，请通知 LFS 开发邮件列表 ，对他们有所帮助。

7.4.3.6. Udev 没有创建设备

后面的文本假设驱动已经被静态的编译进了内核或者是作为模块已经被加载，你已经检查了 Udev 没有创建错误的设备。

某个内核驱动可能没有将其数据导出到 `sysfs` 。这个问题在内核源代码树之外的第三 方驱动程序上尤其常见，结果是这些驱动无法创建其设备节点。用 `/etc/sysconfig/createfiles` 配置文件手动创建这些设备，参考内核文档里的 `devices.txt` 文件或者该驱动的文档以获得正确的主/次设备号。 静态的设备将会被 **S10udev** 启动脚本拷贝到 `/dev` 。

7.4.3.7. 重启后设备名字顺序变化

这是由于 Udev 设计的问题，它以并行方式处理热插拔事件和加载模块。因此名字的顺序就会不可预测。 这将不会被修正。你不应当依赖内核设备名字会稳定。相应的，根据设备稳定的属性（比如，序列号或者 Udev 安装的各种 *_id 工具的输出）写自己 的规则来创建符号链接。可以参见 节 7.12, "为设备创建惯用符号连接" 和 节 7.13, "配置网络脚本" 。

7.5. 配置 setclock 脚本

The `setclock` 脚本从硬件时钟，也就是 BIOS 或 CMOS 时钟读取时间。如果硬件时钟设置为 UTC ，这个脚本会使用 `/etc/localtime` 文件(这个文件把用户所在的时区告诉 `hwclock` 程序)将硬件时钟的时间转换为本地时间。没有办法自动检测硬件时钟是否设置为 UTC 时间，因此需要手动设置。

如果您忘了硬件时钟是不是设置为 UTC 时间了，可以运行 **`hwclock --localtime --show`** 命令，这将显示硬件时钟当前的时间。如果显示的时间符合您的手表的时间，那么硬件时钟设置的是本地时间；如果 `hwclock` 显示的不是本地时间，就有可能设置的是 UTC 时间，可以通过在所显示的 `hwclock` 时间加上或减去您所在时区的小时数来验证。例如，如果您所在的时区是 MST(美国山区时区)，已知是 GMT -0700，在本地时间上加 7 小时。

如果你的硬件使用的不是 UTC 时间，就必须将下面的 `UTC` 变量值设为 `0` (零)，而"UTC=1"表示使用的是UTC时间。

运行下面的命令新建一个 `/etc/sysconfig/clock` 文件：

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# End /etc/sysconfig/clock
EOF
```

在 *http://www.linuxfromscratch.org/hints/downloads/files/time.txt* 有一个很好的关于如何处理 LFS 时间的提示，说明了例如时区、UTC、`TZ` 环境变量等等问题。

7.6. 配置 Linux 控制台

本节讨论如何配置 `console` 初始化脚本来设置键盘映射表和控制台字体。如果您不使用非 ASCII 字符(英镑和欧元符号就是非 ASCII 字符的例子)，并且是美式键盘，可以跳过这一节，没有配置文件的话，`console` 初始化脚本不会做任何事情。

`console` 使用 `/etc/sysconfig/console` 作为配置文件以决定使用什么键盘映射表和屏幕字体，各种特定语言的 HOWTO(参见 *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*) 能帮助您完成配置。LFS-Bootscripts 软件包安装了一个已为一些国家配置好了的预制的 `/etc/sysconfig/console` 文件，如果您所在的国家已经被支持了，您可以去掉相应部分的注释。如果您还有疑问，请在 `/usr/share/kbd` 目录下查找正确的键盘映射表和屏幕字体。 阅读 `loadkeys(1)` 和 `setfont(8)` 的手册，来确定这些程序的正确参数。

`/etc/sysconfig/console 文件中可能包含这样格式的行：` VARIABLE ="value"。变量如下：

KEYMAP

这个变量指定 `loadkeys` 程序的参数。典型的像键盘映射的名字 "es"。 如果不设定参数，bootscript 就不会运行 `loadkeys` 程序, 而是使用默认的内核键盘映射。

KEYMAP_CORRECTIONS

这个参数很少用到，它可被用来指定再次调用 `loadkeys` 程序的参数。对于提供的键盘映射不是很领人满意时，做些调整。 例如，我们要把一些正常情况下不会出现的欧洲字符包含到在键盘映射中, 那我们就需要把这个参数设为 “euro2”。

FONT

这个变量是为 `setfont` 程序设定的. 通常情况下它要包括font的名字, “-m”, 以及需要载入的 application character map名. 比方., 要调用 “lat1-16” font 以及 “8859-1” application character map , 就把这个参数设为 “lat1-16 -m 8859-1”. 如果这个变量没有被设定，bootscript会加载默认的 `setfont` 程序, 以及默认的一大堆

UNICODE

此参数设成“1”, “yes” 或 “true” 会把控制台改为UTF-8模式。在 UTF-8 的 locale 下比较有用，其他情况都是有害的。

LEGACY_CHARSET

对于许多的键盘布局，在 Kbd 包中没有提供 Unicode 键盘映射。如果这个变量被设置为一个有效的非 UTF-8 编码的键盘映射， `console` bootscript 会把它转换成 UTF-8 编码。注意，无效键（例如，键位本身不能产生字符，而是对下一个键产生的字符限制 ;在标准美国键盘中没

有无效键）和组合键（例如，为了产生字符按下 Ctrl+. A E）没有相应的内核 patch 在 UTF-8 模式下是不能工作的。这个变量只用在 UTF-8 模式下。

BROKEN_COMPOSE

如果你应用第八章中的内核 patch ，把它设为 "0"。你不得不在 "-m" 开关之后，添加需要的字符集到 FONT 变量中。这个变量只用在 UTF-8 模式下。

对于把键盘映射直接编译进内核的支持，在内核中已经去掉了。因为这样会导致有一些错误的结果。

一些例子：

- 对于一个非 Unicode 的安装，只有 KEYMAP 和 FONT 变量是必需的。例如，在波兰语安装中，应当使用：

  ```
  cat &gt; /etc/sysconfig/console &lt;&lt; "EOF"# Begin /etc/sysconfig/console

  KEYMAP="pl2"
  FONT="lat2a-16 -m 8859-2"

  # End /etc/sysconfig/console
  EOF
  ```

- 正如上面的方法，有时需要对对提供的键盘映射进行稍微的调整。下面的例子在德国键盘映射中添加欧洲符号：

  ```
  cat &gt; /etc/sysconfig/console &lt;&lt; "EOF"# Begin /etc/sysconfig/console

  KEYMAP="de-latin1"
  KEYMAP_CORRECTIONS="euro2"
  FONT="lat0-16 -m 8859-15"

  # End /etc/sysconfig/console
  EOF
  ```

- 下面是一个保加利亚的 Unicode 例子，存在一个 UTF-8 键盘映射，并且没有定义无效键和组合键规则：

  ```
  cat &gt; /etc/sysconfig/console &lt;&lt; "EOF"# Begin /etc/sysconfig/console

  UNICODE="1"
  KEYMAP="bg_bds-utf8"
  FONT="LatArCyrHeb-16"

  # End /etc/sysconfig/console
  EOF
  ```

- 由于在前面的例子中使用了 512-字符 的 LatArCyrHeb-16 字体，在控制台上不能再显示明亮的颜色，除非使用 framebuffer。如果你想不利用 framebuffer 来显示明亮的颜色，并使用本语种字符，像下面的说明，通过使用相应语言的 256-字符 字体，这仍然是可能实现的。

```
cat &gt; /etc/sysconfig/console &lt;&lt; "EOF"# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- 下面的例子解释了键盘映射从 ISO-8859-15 到 UTF-8 的自动转变和在 Unicode 模式下打开无效键：

```
cat &gt; /etc/sysconfig/console &lt;&lt; "EOF"# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
BROKEN_COMPOSE="0"
FONT="LatArCyrHeb-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- 对于中文、日文、韩文等一些语言，Linux 的控制台不能显示需要的字符。这些语种的用户应当安装 X Window 系统，包含所需字符集的字体，以及合适的输入法（例如，SCIM 支持很多语言）。

注意

`/etc/sysconfig/console` 文件只能控制 Linux 文本控制台。在 X Windows、SSH 会话以及串口控制台中，设置键盘布局和终端字体是没有用的。

7.7. 配置 sysklogd 脚本

`sysklogd` 脚本用 `-m 0` 选项调用 `syslogd` 程序，该选项关闭了周期时间戳标记，这个标记默认让 `syslogd` 每 20 分钟写入一次 log 文件。要打开周期时间戳标记，请编辑 `sysklogd` 脚本做相应修改。请运行 **`man syslogd`** 以获得更多信息。

7.8. 创建 /etc/inputrc 文件

`inputrc` 文件为特定的情况处理键盘映射，这个文件被 Readline 用作启动文件，Readline 是 Bash 和其它大多数 shell 使用的与输入相关的库。

大多数人并不需要自定义键盘映射，所以下面的命令将创建一个适用于所有登陆用户的全局 `/etc/inputrc` 文件。如果你需要为某个用户覆盖默认的设置，你可以在该用户的主目录中创建一个包含自定义键盘映射的 `.inputrc` 文件。

要想了解更多关于如何编辑 `inputrc` 文件的信息，运行 `info bash` 以参考 bash 的 info 页的 *Readline Init File* 这一节，运行 `info readline` 以参考readline 自己的 info 页也不错。

下面是一个基本的全局 `inputrc` 文件，那些选项的注释也一起包括在文件里。请注意，注释不能和命令放在同一行里。

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the
# value contained inside the 1st argument to the
# readline specific functions

"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. Bash Shell 启动文件

shell 程序 `/bin/bash` (在此之后以“shell”称呼)使用了一个启动文件集全，来帮助创造一个运行的环境。每一个文件都有特定的用处，有的文件还能使登入与交互环境有所不同。放在 `/etc` 目录下的一些文件提供了全局设置。如果相类似的设置文件出现在某个用户起始文件夹下(~/)，那么在登入该用户时，它将替代该全局设置。

使用 `/bin/login` 读取 `/etc/passwd` 文件成功登录后，启动了一个交互登录 shell 。用命令行可以启动一个交互非登录 shell(例如 `[prompt]$``/bin/bash`)。非交互 shell 通常出现在 shell 脚本运行的时候，之所以称为非交互的，因为它正在运行一个脚本，而且命令与命令之间并不等待用户的输入。

要获得更多信息，请运行 `info bash` 以参考 *Bash Startup Files and Interactive Shells* 小节。

当以交互登录方式运行 shell 的时候，会读取 `/etc/profile` 和 `~/.bash_profile` 文件。

下面是一个基本的 `/etc/profile` 文件，设置了本地语言支持所必需的环境变量，适当设置这些变量会导致：

- 程序将产生翻译成本地语言的输出
- 正确的将字符分类为字母、数字和其它类，这样做是必要的，可以让 `bash` 在非英语 locale 下正确接收命令行输入的非 ASCII 字符。
- 为指定的国家设置正确的字母排序
- 适当的默认页面大小
- 为货币、时间和日期值设置正确的格式

这个脚本还设置了 `INPUTRC` 环境变量，让 Bash 和 Readline 使用我们先前创建的 `/etc/inputrc` 文件。

把下面的 *`[ll]`* 换成您想要设置的两个字母的语言代码(例如"en")，把 *`[CC]`* 换成适当的两个字母的国家代码(例如"GB")，把 *`[charmap]`* 换成规范的字符映射表。

Glibc 支持的所有 locales 列表可以使用下列命令获得：

```
locale -a
```

很多 locale 有许多别名，比如"ISO-8859-1"也被称为"iso8859-1"和"iso88591"。一些应用程序不能正确的处理这些别名，所以安全的做法是使用 locale 的规范名称。要确定正确的规范名称，运行下面的命令，并把其中的 *`[locale name]`* 替换成 `locale -a` 命令的输出中你感兴趣

的 locale (在这个例子中是"en_GB.iso88591")。

```
LC_ALL=`[locale name]` locale charmap
```

对于"en_GB.iso88591" locale ，上面的命令将会打印：

```
ISO-8859-1
```

这样你就得到了正确的 locale 设置是"en_GB.ISO-8859-1"。将使用上述试探方法得到的 locale 在添加进 Bash 启动文件前做充分的测试是很重要的。

```
LC_ALL=[locale name] locale country
LC_ALL=[locale name] locale language
LC_ALL=[locale name] locale charmap
LC_ALL=[locale name] locale int_curr_symbol
LC_ALL=[locale name] locale int_prefix
```

上述命令将会打印与 *`[locale name]`* 对应的国家和语言名称、使用的字符编码、货币符号、国际长途电话号码前缀。如果上述命令中的任意一个出现了类似下面的错误信息，则表明你该 locale 要么在 Chapter 6 中没有安装，要么不被 Glibc 的默认安装支持。

```
locale: Cannot set LC_* to default locale: No such file or directory
```

在这种情况下，你应当要么使用 `localedef` 命令安装相应的 locale ，要么选择一个其他的 locale 。接下来的指令都将假定 Glibc 没有出现上述错误信息。

一些 LFS 基本系统之外的软件包可能并不支持你选择的 locale 。一个例子就是 X 库(X Window System 的一部分)，它会输出如下信息：

```
Warning: locale not supported by Xlib, locale set to C
```

有时可以通过移除该 locale 的字符映射(charmap)定义部分来解决这个问题——只要这样做不会改变 Glibc 中与该 locale 关联的字符映射关系(这个可以通过对两个 locale 运行 `locale charmap` 命令来检查)。例如将"de_DE.ISO-8859-15@euro"修改为"de_DE@euro"即可让这个 locale 被 Xlib 正确识别。

其他一些软件包在你选择的 locale 出乎其意料的情况下可能即使运行异常也不会显示错误信息。在这种情况下，研究一下其他 Linux 发行版是如何支持你选择的 locale 可能会得到更加有用的信息。

一旦确定了正确的 locale 设置，创建 `/etc/profile` 文件：

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>
export INPUTRC=/etc/inputrc

# End /etc/profile
EOF
```

"C"(默认)和"en_US"(美国英语用户的推荐值)这两个 locale 是不同的。

7.10. 配置 localnet 脚本

`localnet` 脚本的一部分工作是设置系统的主机名，这需要在 `/etc/sysconfig/network` 文件里配置。

运行下面的命令创建 `/etc/sysconfig/network` 文件并设置主机名：

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

`<lfs>` 请用您的计算机名替换 *`[lfs]`* ，不要在这里输入全限定域名(Fully Qualified Domain Name)，FQDN 的信息稍后将放在 `/etc/hosts` 文件里。

7.11. 定制 /etc/hosts 文件

要在 `/etc/hosts` 文件里配置网卡的 IP 地址、FQDN 和可能会用的别名，语法如下：

```
<IP address> myhost.example.org aliases
```

除非您的计算机在 Internet 上是可访问的(例如，有一个注册的域名并分配到了一个合法的 IP 地址(块)(大多数用户没有))，请确保 IP 地址在私有网络 IP 地址范围内，正确的范围是：

```
类别     网络
        A     10.0.0.0
        B     172.16.0.0 到 172.31.0.255
        C     192.168.0.0 到 192.168.255.255
```

一个合法的 IP 地址可能是 192.168.1.1 ，这个 IP 的合法 FQDN 可能是 www.linuxfromscratch.org(不推荐，因为这是个已合法注册的域名，这样做可能会造成域名服务器的问题)。

即使您没有网卡，仍然需要一个 FQDN，某些程序需要这个才能正常工作。

运行下面的命令创建 `/etc/hosts` 文件：

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# End /etc/hosts (network card version)
EOF
```

把 *[192.168.1.1]* 和 *[<HOSTNAME>.example.org]* 更改为特定用户或特别要求所需要的值(如果这台机器要连入一个已存在的网络，并且网络/系统管理员已经给您分配了一个 IP 地址)。

如果您不打算配置网卡，运行下面的命令创建 `/etc/hosts` 文件：

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

7.12. 为设备创建惯用符号连接

7.12.1. CD-ROM symlinks

我们可能装一些软件用到 cdrom dvd 等，因此我们会需要把 /dev/cdrom /dev/dvd 的符号链接加在 `/etc/fstab` 中。对于每一个CD-ROM 设备，在 `/sys 下找到相应的目录` (例如, `/sys/block/hdd`)，然后运行如下命令：

```
udevtest /block/hdd
```

观察一下包含很多 *_id 输出的程序的行。

有两种方法可以创建symlinks，可以用model名及序号，或是用设备在总线上的位置。 以第一种方法，可以创建如下文件：

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF # Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_MODEL}=="SAMSUNG_CD-ROM_SC-148F", \
    ENV{ID_REVISION}=="PS05", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_MODEL}=="PHILIPS_CDD5301", \
    ENV{ID_SERIAL}=="5VO1306DM00190", SYMLINK+="cdrom1 dvd"
EOF
```

注意

这个例子能正常工作，但 udev 不能识别 \ 的继续上一行功能，所以若要用编辑器来编辑 udev 的规则时，一定要保证每行只有一个命令。

做完这些symlinks就会保持正常工作状态，即使把cdrom移到 IDE 总线的其他位置上也能正常工作。但是如果使用新的驱动器来替换原来的 SAMSUNG CD-ROM，`/dev/cdrom 符号链接将不会被创建。`

SUBSYSTEM=="block" 关键字是为了避免匹配一般的 SCSI 设备。 在没有这个关键字的情况下，若同时存在两个 SCSI CD-ROM， 这个符号链接有时会指向 `/dev/srX` 设备 ，但有时会错误的指向 `/dev/sgX` 。

第二种方法的步骤：

```
cat >/etc/udev/rules.d/82-cdrom.rules << EOF # Custom CD-ROM symlinks
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-0:1", SYMLINK+="cdrom"
SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", \
    ENV{ID_PATH}=="pci-0000:00:07.1-ide-1:1", SYMLINK+="cdrom1 dvd"
EOF
```

这样，即使你使用不同的 model 来替换原来的设备，符号链接仍然是正确的，它指向在 IDE 总线上旧的位置。 ENV{ID_TYPE}=="cd" 关键字是为了确保符号链接在总线上的那个位置放的不是 CD-ROM 时，能够消失。

当然把两种方法混合使用也是可以的。

7.12.2. Dealing with duplicate devices

在 节 7.4, "LFS 系统的设备和模块处理"提到过, `/dev` 下相同功能设备的顺序是随机的。例如，你有一个 USB 的网络摄像头和一个 TV 的调谐器，有时 `/dev/video0 指向网络摄像头，``/dev/video1 指向调谐器，但是在重启之后可能就会改变。除了网卡和声卡之外的其` 网卡的解决 方法请见 节 7.13, "配置网络脚本", 声卡解决方法请见 *BLFS*。

每一个设备都可能有这个问题（即使这个问题在你现在的发行版中不存在），在 `/sys/class` 或 `/sys/block 下找到相应的目录。` 对于视频设备，可能是 `/sys/class/video4linux/video_` X `_` 。 找出标记设备唯一性的属性（通常是 设备提供商、产品 ID 以及序列号）：

```
udevinfo -a -p /sys/class/video4linux/video0
```

接下来，写一个创建符号链接的规则，例如：

```
cat >/etc/udev/rules.d/83-duplicate_devs.rules << EOF # Persistent symlinks for webcam
 and tuner
KERNEL=="video*", SYSFS{idProduct}=="1910", SYSFS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", SYSFS{device}=="0x036f", SYSFS{vendor}=="0x109e", \
    SYMLINK+="tvtuner"
EOF
```

结果 `/dev/video0` 和 `/dev/video1` 设备仍然随机指向调谐器和网络摄像头（因此不应当直接使用），但是符号链接 `/dev/tvtuner` 和 `/dev/webcam` 总是指向正确的设备。

关于书写 Udev 规则的更多信息，可以查看 `/usr/share/doc/udev-096/index.html` 。

7.13. 配置网络脚本

本节仅适用于需要配置网卡的情况。

如果不使用网卡，就不需要创建关联网卡的配置文件，这样的话，在所有运行级目录(`/etc/rc.d/rc*.d`) 下删除 `network` 符号链接。

7.13.1. 创建网络接口的稳定名称

这一段的说明对于单网卡是可选的。

由于 Udev 和 网络驱动的模块化，网络设备的接口的加载顺序在每次reboot后可能会不同，因为驱动是并行加载的，所以顺序会变成随机。例如，在一台计算机上有两块网 卡Intel 和Realtek。Intel 制造的网卡可能是 eth0，Realtelk 的网卡是 eth1;但是重启后网卡的顺序可能反过来。为避免这种情况，我们应该 根据网卡的MAC地址或总线位置来为他们命名。

如果想要根据MAC地址来识别网卡，可以使用如下命令：

```
grep -H . /sys/class/net/*/address
```

为每个网卡（除loopback），设计一个描述性的名字，比方 “realtek”, 然后参照如下建立Udev规则：

```
cat > /etc/udev/rules.d/26-network.rules << EOFACTION=="add", SUBSYSTEM=="net", SYSFS{
address}=="_`00:e0:4c:12:34:56`_", \
    NAME="_`realtek`_"
ACTION=="add", SUBSYSTEM=="net", SYSFS{address}=="_`00:a0:c9:78:9a:bc`_", \
    NAME="_`intel`_"
EOF
```

注意

虽然这个例子可以正常工作，但要注意 udev 不能识别 \ 的继续上一行的功能。所以如果用文本编辑器来编辑就一定要保证每个规则占一行。

如果要以总线位置作为标准，可以创建如下的Udev规则：

```
cat > /etc/udev/rules.d/26-network.rules << EOFACTION=="add", SUBSYSTEM=="net", BUS=="
_`pci`_", ID=="_`0000:00:0c.0`_", \
    NAME="_`realtek`_"
ACTION=="add", SUBSYSTEM=="net", BUS=="_`pci`_", ID=="_`0000:00:0d.0`_", \
    NAME="_`intel`_"
EOF
```

这个规则会把网卡名字每次都定为 “realtek” 和 “intel”, 以替代eth0和eth1 （即：原来的eth0和eth1接口不存在了 除非把他们加到 NAME 下）。在下面的文件中我们使用描述性的名字来替代eth0 等。

需注明以上规则并不永远适用。例如, 当VLAN 和 网桥 使用时，以MAC为基准的命名就会出问题。由于 VLAN 和 网桥 在网卡上有相同的MAC地址。 如果只想重新命名网卡接口，而不是brige和VLAN接口，但是规则都会匹配它们。如果使用虚拟接口，我们会有两种解决办法。其一是把 DRIVER=="?*" 关键字加在 SUBSYSTEM=="net" 后。但这种方法对于一些老型号的网卡不起作用，因为这些网卡的 uevent 中没有 DRIVER 变量，因此 按规则就不可能匹配到这些卡。 另外一种就是以总线位置命名。

另外一种已知的不能工作的情况存在于无线网络中，当应用MadWifi 或 HostAP 驱动时, 他们会以相同的MAC地址和总线位置创建至少两个接口。例如，Madwifi 驱动会创建一个 athX 和 wifiX 接口（X是一个数字）。为区别 这些接口，可以把 KERNEL 参数，比如 KERNEL=="ath*" 加到 SUBSYSTEM=="net"后。

可能还有很多情况会导致不能正常工作，现在这方面的 bug 仍然不断的报告给 Linux 的各个发布版，没有一个解决方法可以解决所有的情况。

7.13.2. 创建网络接口配置文件

network 脚本启用或关闭哪个接口由 `/etc/sysconfig/network-devices` 目录下的文件决定，这个目录下的文件应该是类似于 `ifconfig.xyz` 的形式，这里"xyz"是网络接口名(例如 eth0 或者 eth0:1)。这个目录中的文件将定义接口的属性，比如IP地址、子网掩码等等。

在这个目录下新建文件，下面是一个为 *eth0* 设备创建 `ipv4` 文件的示例：

```
cd /etc/sysconfig/network-devices &&mkdir -v ifconfig.eth0 &&cat > ifconfig.eth0/ipv4
<< "EOF"ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

每个文件中的这些变量的值都要改成您的设置，如果 `ONBOOT` 变量设置为"yes"，network 脚本会在系统启动的时候启动 NIC(Network Interface Card 网络接口卡，简称网卡)，如果设置为"yes"以外的值，网卡会被 network 脚本忽略而没有启动。

`SERVICE` 变量定义获取 IP 地址的方式，LFS-Bootscripts 有一套模块化的 IP 地址分配格式，并在 `/etc/sysconfig/network-devices/services` 目录下为其它的 IP 分配方式创建了附加的文件，这通常用作 DHCP(Dynamic Host Configuration Protocol 动态主机配置协议)方式，在 BLFS 里有详细介绍。

`GATEWAY` 变量应该设置为默认网关的 IP 地址，如果没有默认网关，就把这个变量完全注释掉。

`PREFIX` 变量设置为子网使用的位数，IP 地址的每个字节是 8 bit ，如果子网掩码是 255.255.255.0 ，那么它使用前三个字节(24 bit)指定网络号；如果网络掩码是 255.255.255.240 ，它用前 28 bit 来指定网络号。长于 24 bit 的前缀一般由 DSL 和 cable 的 ISP(Internet Service Providers 因特网服务提供商)使用，我们的例子里(PREFIX=24)，子网掩码是 255.255.255.0 ，请根据您的网络情况调整 `PREFIX` 变量。

7.13.3. 创建 /etc/resolv.conf 文件

如果系统要连接到 Internet 上，就需要 DNS(Domain Name Service 域名服务)名称解析的手段，来把 Internet 域名解析为 IP 地址，反之亦然。在 `/etc/resolv.conf` 文件里设置 ISP 或网络管理员提供的域名服务器的 IP 地址就可以达到这个目的了，运行下面的命令创建这个文件：

```
cat > /etc/resolv.conf << "EOF"# Begin /etc/resolv.conf

domain {<域名>}
nameserver <主域名服务器IP地址>
nameserver <副域名服务器IP地址>

# End /etc/resolv.conf
EOF
```

把 `[域名服务器IP地址]` 替换为您的域名服务器的 IP 地址。域名服务器常常不止一项(作为备份用途)，如果您只需要一个域名服务器，把文件里的第二行 *nameserver* 删除就可以了。在局域网里这个 IP 地址还可能是路由器。

8. 使 LFS 系统能够启动

8.1. 简介

此时可以让 LFS 启动了，本章节讨论创建 `fstab` 文件, 为新的 LFS 系统编译一个内核，并安装 Grub 引导程序，在启动菜单上选择 LFS 系统后能启动系统。

8.2. 创建 /etc/fstab 文件

一些程序用 `/etc/fstab` 文件来确定哪一些文件系统是默认被加载了，加载顺序情况，哪些必须被检查的(完整性错误校验)。创建一个新的文件系统表大致如下所示：

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type   options         dump  fsck
#                                                        order

/dev/<xxx>     /            <fff>  defaults        1     1
/dev/<yyy>     swap         swap   pri=1           0     0
proc           /proc        proc   defaults        0     0
sysfs          /sys         sysfs  defaults        0     0
devpts         /dev/pts     devpts gid=4,mode=620  0     0
shm            /dev/shm     tmpfs  defaults        0     0
# End /etc/fstab
EOF
```

在你的系统上替换 `<xxx>` ， `<yyy>` ，和 `<fff>` 为适当的值，例如， `hda2` , `hda5` ，and `ext3` 。 关于文件中六个字段的详细信息，可通过 `man 5 fstab` 获取。

这个 `/dev/shm` 挂载点是为 `tmpfs` (虚拟内存文件系统)能包括启用POSIX共享内存。这需要内核必须在构建的时候支持这个选项才能起作用(更多相关信息在下一个章节)。请注意，目前很少软件使用POSIX共享内存。 所以，认为 `/dev/shm` mount 挂载点选项是可选择的，更多信息请查看内核源码树里的 `Documentation/filesystems/tmpfs.txt` 。

文件系统有 MS-DOS 或 Windows 血统(i.e.: vfat，ntfs，smbfs, cifs，iso9660，udf)需要有"iocharset" 加载选项载，以使文件名称的非ASCII(美国信息交换标准代码)特征被适当解释。这个选项和你的所处位置特征是一样的，用这个方式调整内核支持它。如果需要操作相关的特征定义(在 File systems -> Native Language Support下可找到)，可以编译进内核里或编译成模块。 "codepage" 选项同样也是为 vfat 和 smbfs 文件系统所需要. 它应该被设置为 MS-DOS 在你的国家里使用的 codepage 数字号。举个例子，为了挂在 USB flash 设备，ru_RU.KOI8-R 的使用者需要在 `/etc/fstab` 里的以下行:

```
/dev/sda1     /media/flash vfat
    noauto,user,quiet,showexec,iocharset=koi8r,codepage=866 0 0
```

ru_RU.UTF-8 使用者的相应行：

```
/dev/sda1     /media/flash vfat
    noauto,user,quiet,showexec,iocharset=utf8,codepage=866 0 0
```

注意

在后面的例子里，内核发出如下信息：

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
    filesystem will be case sensitive!
```

这个否定的建议应该可以被忽略，因为所有的其他"iocharset" 值选项造成UTF-8 locales文件名的错误显示。

它也可能是在内核构造的时候为一些文件系统指定默认的 codepage 和 iocharset。相关的参数是指定的 "默认NLS选项" (CONFIG_NLS_DEFAULT)，"默认远程NLS选项" (CONFIG_SMB_NLS_DEFAULT)，"默认的FAT的codepage " (CONFIG_FAT_DEFAULT_CODEPAGE)，和 "默认FAT的iocharset" (CONFIG_FAT_DEFAULT_IOCHARSET)。在这里不叙述在内核编译时为 ntfs 文件的系统设置。

8.3. Linux-2.6.16.27

Linux 内核软件包包含内核源代码及其头文件。

预计编译时间： 1.5 - 3 SBU所需磁盘空间： 310 - 350 MB

8.3.1. 安装 kernel

编译内核包含几个步骤——配置、编译和安装。阅读内核源码树里的 `README` 文件可选择不同于本书的其他配置内核方法。

预先设定情况下，当在 UTF-8 键盘模式里，键没反应，是Linux 内核发生的字节顺序错误 。同样，在 UTF-8 模式起作用的情况下，有一个不能拷贝和粘贴非ASCII的特征。用发布的补丁可修复：

```
patch -Np1 -i ../linux-2.6.16.27-utf8_input-1.patch
```

运行下面的命令准备编译：

```
make mrproper
```

这能保证内核树绝对干净。内核开放团队推荐每次编译内核之前都运行这个命令。不要相信解压后的源码树就是干净的。

用菜单形式界面的配置内核。BLFS 有一些关于配置内核必备条件的细节，不在 LFS 包里，在*http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index*:

```
make menuconfig
```

本文译者也有一篇文章，《Linux-3.10-x86_64 内核配置选项简介》，介绍了几乎每一个内核选项。对于使用 VMware 虚拟机的读者，这里还有一份基于 2.6.24 内核的 .config 文件可供参考(切忌照搬照抄,因为这样是行不通的)。

选择 **make oldconfig** 在同样情况下，可能会更适合。阅读 `README` 可获取更多信息。

如果愿意，你可以跳过内核配置，直接复制内核配置文件 `.config` ,从主机系统(如果可用)解压到 `linux-2.6.16.27` 目录。当然，我们不推荐这个方式。通常探究所有配置菜单项，再根据需要创建内核是最佳的。

编译内核镜像和模块：

```
make
```

如果使用内核模块，`/etc/modprobe.conf` 文件是必须的。有关模块和内核配置的信息在 7.4, "LFS 系统的设备和模块处理"，还有 `linux-2.6.16.27/Documentation` 目录中的内核文档。同样，引起注意的可能还有 `modprobe.conf(5)` 。

安装模块，如果内核配置使用它们：

```
make modules_install
```

内核编译完成后，增加步骤完成安装是必须的。一些文件需要拷贝副本到 `/boot` 目录。

内核镜像的路径，根据不同的平台可能会改变，下面的命令假定在x86架构上：

```
cp -v arch/i386/boot/bzImage /boot/lfskernel-2.6.16.27
```

`System.map` 是一个内核的符号文件 。它映射每个内核API函数的入口，以及内核在运行中内核数据结构的地址。 运行下面这个命令安装这个文件：

```
cp -v System.map /boot/System.map-2.6.16.27
```

内核配置文件 `.config` 产生于**make menuconfig** 这个步骤，包含所有的内核配置选择被编译。一个好主意是保留这个文件以备将来参考：

```
cp -v .config /boot/config-2.6.16.27
```

安装Linux内核文档：

```
install -d /usr/share/doc/linux-2.6.16.27 &&
cp -r Documentation/* /usr/share/doc/linux-2.6.16.27
```

有一点重要提示，内核源码目录的所有者不是 *root*。只要是用 *root* (类似我们在 chroot环境里)，解压出来的文件不是计算机上的用户和组 ID，对于其他包这不是问题，在安装后被删除源码树。但是，Linux源码通常保留很长时间。因为 碰巧有一个用户 ID 和这个软件打包者的用户组 ID 相同，那么他可以拥有对内核源码的写权限。

如果这个内核源码准备保留，在 `linux-2.6.16.27` 目录运行 `chown -R 0:0` ，确保所有文件的属主是 *root*.

警告

一些内核文档推荐建立一个 `/usr/src/linux` 的链接指向内核源码目录。这个是对2.6版本内核的要求，而且在 LFS 系统上 不允许 ，它会导致你可能在完成LFS系统构建后，再安装其他软件包出现错误。

同样，系统的 `include` 目录下的头文件应该 永远 保持Glibc编译后的版本， 和 Linux-Libc-Headers 包相同，所以应该要 决不 替换内核的头文件。

8.3.2. Linux 的内容

安装的文件： config-2.6.16.27, lfskernel-2.6.16.27, System.map-2.6.16.27

简要描述

`config-2.6.16.27`	包含所有内核配置选择后的项
`lfskernel-2.6.16.27`	Linux 系统的引擎。当启动计算机时，内核是操作系统装载的第一个部分，它检测并初始化所有的电脑硬件的组件，然后将这些设备以文件树的形式存放使得其它软件可以访问，并且能让单个 CPU 成为多任务处理机器能力，可以同时运行许多程序。
`System.map-2.6.16.27`	显示地址和符号的文件；它映射内核里所有函数和数据结构的入口和地址。

8.4. 使 LFS 系统能够启动

你的全新 LFS 系统差不多要完成了。 最后要做的事是确保系统可以正常启动。下面的指令仅适用于 IA-32 架构的计算机，就是主流的 PC 机。 关于其它架构计算机 "boot loading（引导装载）"的信息可以在相应的资源里找到。

引导装载是一个很复杂的问题，因此接下来会有一些警告的话。所以需要熟悉当前的引导装置和硬盘上其他操作系统需要能被启动。确定紧急启动盘已经准备了，假如电脑变成不能用了（不能启动）能够 "援救"。

先前，我们为这个步骤编译和安装了 GRUB 引导装载程序做了准备。这个程序包括了在硬盘的指定位置上写的一些特殊 GRUB 文件，我们强烈推荐你创建一张 GRUB 引导软盘作为备份，插入一张空白软盘并输入下面的命令：

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

取出软盘并在安全的地方存放，现在，运行 `grub` shell：

```
grub
```

GRUB 使用它自己的驱动器和分区命名结构，形式是 *(hdn,m)*，这里的 *n* 是硬盘驱动号， *m* 是分区号， 两个都是从零开始。例如，分区 `hda1` 是GRUB的 *(hd0,0)* ， `hdb3` 是 *(hd1,2)*. 与 Linux 不同的是， GRUB 不把光驱作为硬盘驱动器。 例如，假如 `hdb` 是光盘驱动器，第二个硬盘驱动器是 `hdc` ，第二个硬盘驱动器仍然是 *(hd1)*。

用上面的信息为 root 分区（或boot 分区，假如是单独使用了分区的情况下）。 下面的例子里假定 root 分区（或单独的 boot分区）是 `hda4` .

告诉 GRUB 在哪里搜索它的 `stage{1,2}` 文件。用 Tab 键能在各处让 GRUB 显示可选择项：

```
root (hd0,3)
```

警告

下一个命令会覆盖当前的引导装载程序，如果不需要的话就不要运行这个命令，例如，使用第三方启动管理器来管理主引导记录 (MBR)。当然，现在的情况是安装 GRUB 到 LFS 分区的“boot sector”更有意义。在这个例子里，下一个命令将变成 **`setup (hd0,3)`** 。

告诉 GRUB 安装它自己到 `hda` 的MBR：

```
setup (hd0)
```

如果一切顺利，GRUB 会报告在 `/boot/grub` 找到它的文件。现在可以退出 `grub` shell：

```
quit
```

创建一个"显示菜单"文件定义 GRUB 的启动菜单：

```
cat > /boot/grub/menu.lst << "EOF"
# Begin /boot/grub/menu.lst

# By default boot the first menu entry.
default 0

# Allow 30 seconds before booting the default.
timeout 30

# Use prettier colors.
color green/black light-green/black

# The first entry is for LFS.
title LFS 6.2
root (hd0,3)
kernel /boot/lfskernel-2.6.16.27 root=/dev/hda4
EOF
```

如果需要可以为宿主系统增加一项，看起来如下：

```
cat >> /boot/grub/menu.lst << "EOF"
title Red Hat
root (hd0,2)
kernel /boot/kernel-2.6.5 root=/dev/hda3
initrd /boot/initrd-2.6.5
EOF
```

如果是 Windows 的双启动系统，下面的项能够启动它：

```
cat >> /boot/grub/menu.lst << "EOF"
title Windows
rootnoverify (hd0,0)
chainloader +1
EOF
```

如果用 `info grub` 不能获取足够的信息，更多 GRUB 资料可以在它的网站找到 http://www.gnu.org/software/grub/.

FHS 规定 GRUB 的 `menu.lst` 文件必须链接到 `/etc/grub/menu.lst`。为了符合这个规定，可以用下面的命令：

```
mkdir -v /etc/grub &&
ln -sv /boot/grub/menu.lst /etc/grub
```

9. 结束

9.1. 结束

顺利完成了！新 LFS 系统已经安装完成！我们祝贺你在闪亮的自定义的 Linux 系统上有更多的成功。

创建一个 `/etc/lfs-release` 文件是个好主意。有了这个文件，你能方便地在系统上(和告诉我们你需要询问的帮助指向)找到 LFS 是哪个版本。运行下面语句创建文件：

```
echo 6.2 > /etc/lfs-release
```

9.2. 看看你是第几个?

现在你已经完成本书全部步骤，你想成为 LFS 计算内的用户吗？ 从 *http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi* 上输入你的姓名和你使用的第一个 LFS 版本注册为 LFS 用户。

现在让我们重启进入 LFS 系统。

9.3. 重启系统

现在所有的软件都安装完毕，到了重启你的电脑的时候了。但是，你应该要注意一些事。你用这本书创建的系统是相当最小限度的系统， 你很可能缺少需要的很多连续前进的功能。用一些来自 BLFS 书额外的包，在最近仍然使用的 chroot 环境中安装，你能离你自己的宿主机器，在你重启一次后进入新的 LFS 系统，可以用更多更佳的形式来继续安装。安装一个文本模式的浏览器，例如象 Lynx，你能在一个虚拟终端方便地观看 BLFS 书，安装包在另一个终端上。GPM 包将同样允许你在你地虚拟终端上完成拷贝/粘贴动作 最后，假如如果静态 IP配置不能符合你的网络需求条件，安装 Dhcpcd 或 PPP 包，这点也是同样有用处的。

现在我们说完了，离开我们闪亮的新 LFS 安装，第一次启动！首先，退出 chroot 环境：

```
logout
```

卸载虚拟文件系统：

```
umount -v $LFS/dev/pts
umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

卸载 LFS 自己的文件系统：

```
umount -v $LFS
```

如果多个分区被创建，卸载其他分区之前先卸载主要的一个，比如这个：

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

现在，重启系统：

```
shutdown -r now
```

早先的概要说明时，假如 GRUB 引导装载程序已经设置，按照启动菜单的设置可自动启动 *LFS 6.2* 。

当重启完毕，LFS 系统已经准备使用更多你需要增加的程序。

9.4. 现在做什么?

感谢你阅读 LFS 书。我们希望你能在书中找到帮助和学习更多关于系统创建过程的知识。

现在 LFS 已经安装完毕，你可能觉得比较疑惑"下一步怎么做？"为了回答这问题，我们编制了一个资源列表给你。

- 维护

  所有软件的 Bugs 和安全通告。既然一个 LFS 系统从源代码中构建成，跟踪报告并更新软件取决你自己。这里有 几个在线资源跟踪报告，其中一些在下面：

  - Freshmeat.net (http://freshmeat.net/)

    Freshmeat 能提醒你(通过 email)，在你的系统上最新安装包的新版本。

  - CERT (Computer Emergency Response Team)

    CERT 是一个发布关于不同操作系统和应用程序的安全警告邮件列表。 订阅有效的信息在 http://www.us-cert.gov/cas/signup.html.

  - Bugtraq

    Bugtraq 是一个完全揭露计算机安全的邮件列表。它发布最新发现的安全发布公告，并偶尔提供为其修复的方法。有效发布信息在http://www.securityfocus.com/archive.

- Beyond Linux From Scratch

  Beyond Linux From Scratch 书涵盖了 LFS 书之外的大量范围的软件安装。BLFS 项目在 http://www.linuxfromscratch.org/blfs/.

- LFS Hints

  LFS Hints 收藏了 LFS 社区自愿者提交的文档，hints 可用到的在http://www.linuxfromscratch.org/hints/list.html.

- 邮件列表

  这是几个 LFS 邮件列表，假如你需要帮助可以订阅当前的开放，或者要为这个项目贡献力量和其他，查看 第一章节 得到更多信息。

- Linux文档项目

  Linux 文档项目的目的 (TLDP)是收集所有 Linux 发行版文档。 TLDP 的特点是收集了大量的 HOWTOs、guides,和 man pages。这个计划在 http://www.tldp.org/.

IV. 附录

A. 缩写和名词

ABI	Application Binary Interface
ALFS	Automated Linux From Scratch
ALSA	Advanced Linux Sound Architecture
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input/Output System
BLFS	Beyond Linux From Scratch
BSD	Berkeley Software Distribution
chroot	change root
CMOS	Complementary Metal Oxide Semiconductor
COS	Class Of Service
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CVS	Concurrent Versions System
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
EGA	Enhanced Graphics Adapter
ELF	Executable and Linkable Format
EOF	End of File
EQN	equation
EVMS	Enterprise Volume Management System
ext2	second extended file system
ext3	third extended file system
FAQ	Frequently Asked Questions
FHS	Filesystem Hierarchy Standard
FIFO	First-In, First Out
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol

GB	Gibabytes
GCC	GNU Compiler Collection
GID	Group Identifier
GMT	Greenwich Mean Time
GPG	GNU Privacy Guard
HTML	Hypertext Markup Language
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronic Engineers
IO	Input/Output
IP	Internet Protocol
IPC	Inter-Process Communication
IRC	Internet Relay Chat
ISO	International Organization for Standardization
ISP	Internet Service Provider
KB	Kilobytes
LED	Light Emitting Diode
LFS	Linux From Scratch
LSB	Linux Standard Base
MB	Megabytes
MBR	Master Boot Record
MD5	Message Digest 5
NIC	Network Interface Card
NLS	Native Language Support
NNTP	Network News Transport Protocol
NPTL	Native POSIX Threading Library
OSS	Open Sound System
PCH	Pre-Compiled Headers
PCRE	Perl Compatible Regular Expression
PID	Process Identifier
PLFS	Pure Linux From Scratch
PTY	pseudo terminal
QA	Quality Assurance

QOS	Quality Of Service
RAM	Random Access Memory
RPC	Remote Procedure Call
RTC	Real Time Clock
SBU	Standard Build Unit
SCO	The Santa Cruz Operation
SGR	Select Graphic Rendition
SHA1	Secure-Hash Algorithm 1
SMP	Symmetric Multi-Processor
TLDP	The Linux Documentation Project
TFTP	Trivial File Transfer Protocol
TLS	Thread-Local Storage
UID	User Identifier
umask	user file-creation mask
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
VC	Virtual Console
VGA	Video Graphics Array
VT	Virtual Terminal

B. 致谢

We would like to thank the following people and organizations for their contributions to the Linux From Scratch Project.

- *Gerard Beekmans* <gerard AT linuxfromscratch D0T org> – LFS Creator, LFS Project Leader
- *Matthew Burgess* <matthew AT linuxfromscratch D0T org> – LFS Project Leader, LFS Technical Writer/Editor, LFS Release Manager
- *Archaic* <archaic AT linuxfromscratch D0T org> – LFS Technical Writer/Editor, HLFS Project Leader, BLFS Editor, Hints and Patches Project Maintainer
- *Nathan Coulson* <nathan AT linuxfromscratch D0T org> – LFS-Bootscripts Maintainer
- *Bruce Dubbs* <bdubbs AT linuxfromscratch D0T org> – BLFS Project Leader
- *Manuel Canales Esparcia* <manuel AT linuxfromscratch D0T org> – LFS/BLFS/HLFS XML and XSL Maintainer
- *Jim Gifford* <jim AT linuxfromscratch D0T org> – LFS Technical Writer, Patches Project Leader
- *Jeremy Huntwork* <jhuntwork AT linuxfromscratch D0T org> – LFS Technical Writer, LFS LiveCD Maintainer, ALFS Project Leader
- *Anderson Lizardo* <lizardo AT linuxfromscratch D0T org> – Website Backend-Scripts Maintainer
- *Ryan Oliver* <ryan AT linuxfromscratch D0T org> – LFS Toolchain Maintainer
- *James Robertson* <jwrober AT linuxfromscratch D0T org> – Bugzilla Maintainer
- *Tushar Teredesai* <tushar AT linuxfromscratch D0T org> – BLFS Book Editor, Hints and Patches Project Leader
- Countless other people on the various LFS and BLFS mailing lists who helped make this book possible by giving their suggestions, testing the book, and submitting bug reports, instructions, and their experiences with installing various packages.

Translators

- *Manuel Canales Esparcia* <macana AT macana-es D0T com> – Spanish LFS translation project
- *Johan Lenglet* <johan AT linuxfromscratch D0T org> – French LFS translation project
- *Anderson Lizardo* <lizardo AT linuxfromscratch D0T org> – Portuguese LFS translation project
- *Thomas Reitelbach* <tr AT erdfunkstelle D0T de> – German LFS translation project

Mirror Maintainers

North American Mirrors

- *Scott Kveton* <scott AT osuosl D0T org> – lfs.oregonstate.edu mirror
- *Mikhail Pastukhov* <miha AT xuy D0T biz> – lfs.130th.net mirror
- *William Astle* <lost AT l-w D0T net> – ca.linuxfromscratch.org mirror
- *Jeremy Polen* <jpolen AT rackspace D0T com> – us2.linuxfromscratch.org mirror
- *Tim Jackson* <tim AT idge D0T net> – linuxfromscratch.idge.net mirror
- *Jeremy Utley* <jeremy AT linux-phreak D0T net> – lfs.linux-phreak.net mirror

South American Mirrors

- *Andres Meggiotto* <sysop AT mesi D0T com D0T ar> – lfs.mesi.com.ar mirror
- *Manuel Canales Esparcia* <manuel AT linuxfromscratch D0T org> – lfsmirror.lfs-es.info mirror
- *Eduardo B. Fonseca* <ebf AT aedsolucoes D0T com D0T br> – br.linuxfromscratch.org mirror

European Mirrors

- *Barna Koczka* <barna AT siker D0T hu> – hu.linuxfromscratch.org mirror
- *UK Mirror Service* – linuxfromscratch.mirror.ac.uk mirror
- *Martin Voss* <Martin D0T Voss AT ada D0T de> – lfs.linux-matrix.net mirror
- *Guido Passet* <guido AT primerelay D0T net> – nl.linuxfromscratch.org mirror
- *Bastiaan Jacques* <baafie AT planet D0T nl> – lfs.pagefault.net mirror

- *Roel Neefs* <lfs-mirror AT linuxfromscratch D0T rave D0T org> – linuxfromscratch.rave.org mirror
- *Justin Knierim* <justin AT jrknierim D0T de> – www.lfs-matrix.de mirror
- *Stephan Brendel* <stevie AT stevie20 D0T de> – lfs.netservice-neuss.de mirror
- *Antonin Sprinzl* <Antonin D0T Sprinzl AT tuwien D0T ac D0T at> – at.linuxfromscratch.org mirror
- *Fredrik Danerklint* <fredan-lfs AT fredan D0T org> – se.linuxfromscratch.org mirror
- *Parisian sysadmins* <archive AT doc D0T cs D0T univ-paris8 D0T fr> – www2.fr.linuxfromscratch.org mirror
- *Alexander Velin* <velin AT zadnik D0T org> – bg.linuxfromscratch.org mirror
- *Dirk Webster* <dirk AT securewebservices D0T co D0T uk> – lfs.securewebservices.co.uk mirror
- *Thomas Skyt* <thomas AT sofagang D0T dk> – dk.linuxfromscratch.org mirror
- *Simon Nicoll* <sime AT dot-sime D0T com> – uk.linuxfromscratch.org mirror

Asian Mirrors

- *Pui Yong* <pyng AT spam D0T averse D0T net> – sg.linuxfromscratch.org mirror
- *Stuart Harris* <stuart AT althalus D0T me D0T uk> – lfs.mirror.intermedia.com.sg mirror

Australian Mirrors

- *Jason Andrade* <jason AT dstc D0T edu D0T au> – au.linuxfromscratch.org mirror

Former Project Team Members

- *Christine Barczak* <theladyskye AT linuxfromscratch D0T org> – LFS Book Editor
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen AT linuxfromscratch D0T org> – Website Developer, FAQ Maintainer

- Alex Groenewoud – LFS Technical Writer
- Marc Heerdink
- Mark Hymers
- Seth W. Klein – FAQ maintainer
- *Nicholas Leippe* <nicholas AT linuxfromscratch D0T org> – Wiki Maintainer
- Simon Perreault
- *Scot Mc Pherson* <scot AT linuxfromscratch D0T org> – LFS NNTP Gateway Maintainer
- *Alexander Patrakov* <semzx AT newmail D0T ru> – LFS Technical Writer
- *Greg Schafer* <gschafer AT zip D0T com D0T au> – LFS Technical Writer
- Jesse Tie-Ten-Quee – LFS Technical Writer
- *Jeremy Utley* <jeremy AT linuxfromscratch D0T org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer
- *Zack Winkles* <zwinkles AT gmail D0T com> – LFS Technical Writer

A very special thank you to our donators

- *Dean Benson* <dean AT vipersoft D0T co D0T uk> for several monetary contributions
- *Hagen Herrschaft* <hrx AT hrxnet D0T de> for donating a 2.2 GHz P4 system, now running under the name of Lorien
- *SEO Company Canada* supports Open Source projects and different Linux distributions
- *VA Software* who, on behalf of *Linux.com*, donated a VA Linux 420 (former StartX SP2) workstation
- Mark Stone for donating Belgarath, the linuxfromscratch.org server

C. 依赖关系

Every package built in LFS relies on one or more other packages in order to build and install properly. Some packages even participate in circular dependencies, that is, the first package depends on the second which in turn depends on the first. Because of these dependencies, the order in which packages are built in LFS is very important. The purpose of this page is to document the dependencies of each package built in LFS.

For each package we build, we have listed three types of dependencies. The first lists what other packages need to be available in order to compile and install the package in question. The second lists what packages, in addition to those on the first list, need to be available in order to run the testsuites. The last list of dependencies are packages that require this package to be built and installed in its final location before they are built and installed. In most cases, this is because these packages will hardcode paths to binaries within their scripts. If not built in a certain order, this could result in paths of /tools/bin/[binary] being placed inside scripts installed to the final system. This is obviously not desirable.

Autoconf

Installation depends on: Bash, Coreutils, Grep, M4, Make, Perl, Sed, Texinfo**Test suite depends on:** Automake, Diffutils, Findutils, GCC, Libtool**Must be installed before:** Automake

Automake

Installation depends on: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, Texinfo**Test suite depends on:** Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, Tar. Can also use several other packages that are not installed in LFS.**Must be installed before:** None

Bash

Installation depends on: Bash, Bison, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, Texinfo**Test suite depends on:** Diffutils, Gawk**Must be installed before:** None

Berkeley DB

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed**Test suite depends on:** Not run. Requires TCL installed on the final system**Must be installed before:** None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl, Sed, Texinfo**Test suite depends on:** DejaGNU, Expect**Must be installed before:** None

Bison

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Sed**Test suite depends on:** Diffutils and Findutils**Must be installed before:** Flex, Kbd, Tar

Bzip2

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, Patch**Test suite depends on:** None**Must be installed before:** None

Coreutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, Texinfo**Test suite depends on:** Diffutils**Must be installed before:** Bash, Diffutils, Findutils, Man-DB, Udev

DejaGNU

Installation depends on: Bash, Coreutils, Diffutils, GCC, Grep, Make, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Diffutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, Texinfo**Test suite depends on:** No testsuite available**Must be installed before:** None

Expect

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, Tcl**Test suite depends on:** None**Must be installed before:** None

E2fsprogs

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, Make, Sed, Texinfo**Test suite depends on:** Diffutils**Must be installed before:** Util-Linux

File

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Zlib**Test suite depends on:** No testsuite available**Must be installed before:** None

Findutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, Texinfo**Test suite depends on:** DejaGNU, Diffutils, Expect**Must be installed before:** None

Flex

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, Texinfo**Test suite depends on:** Bison, Gawk**Must be installed before:** IPRoute2, Kbd, Man-DB

Gawk

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and, Texinfo**Test suite depends on:** Diffutils**Must be installed before:** None

Gcc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, Tar, Texinfo**Test suite depends on:** DejaGNU, Expect**Must be installed before:** None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, Texinfo**Test suite depends on:** Diffutils, Perl, Tcl**Must be installed before:** Automake

Glibc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed, Texinfo**Test suite depends on:** None**Must be installed before:** None

Grep

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Patch, Sed, Texinfo**Test suite depends on:** Diffutils, Gawk**Must be installed before:** Man-DB

Groff

Installation depends on: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, Texinfo**Test suite depends on:** No testsuite available**Must be installed before:** Man-DB, Perl

GRUB

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Sed, Texinfo**Test suite depends on:** None**Must be installed before:** None

Gzip

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, Texinfo**Test suite depends on:** No testsuite available**Must be installed before:** Man-DB

Iana-Etc

Installation depends on: Coreutils, Gawk, Make**Test suite depends on:** No testsuite available**Must be installed before:** Perl

Inetutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo**Test suite depends on:** No testsuite available**Must be installed before:** Tar

IProute2

Installation depends on: Bash, Berkeley DB, Bison, Coreutils, Flex, GCC, Glibc, Make, Linux-Libc-Headers**Test suite depends on:** No testsuite available**Must be installed before:** None

Kbd

Installation depends on: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Less

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Libtool

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Texinfo**Test suite depends on:** Findutils**Must be installed before:** None

Linux Kernel

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Module-Init-Tools, Ncurses, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

M4

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed**Test suite depends on:** Diffutils**Must be installed before:** Autoconf and Bison

Man-DB

Installation depends on: Bash, Berkeley DB, Binutils, Bzip2, Coreutils, Flex, GCC, Gettext, Glibc, Grep, Groff, Gzip, Less, Make, Sed**Test suite depends on:** Not run. Requires Man-DB testsuite package**Must be installed before:** None

Make

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Sed, Texinfo**Test suite depends on:** Perl**Must be installed before:** None

Mktemp

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Patch, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Module-Init-Tools

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, Zlib**Test suite depends on:** File, Findutils, Gawk**Must be installed before:** None

Ncurses

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed**Test suite depends on:** No testsuite available**Must be installed before:** Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-Linux, Vim

Patch

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Perl

Installation depends on: Bash, Berkeley DB, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Groff, Make, Sed**Test suite depends on:** Iana-Etc, Procps**Must be installed before:** Autoconf

Procps

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make, Ncurses**Test suite depends on:** No testsuite available**Must be installed before:** None

Psmisc

Installation depends on: Bash, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Readline

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo**Test suite depends on:** No testsuite available**Must be installed before:** Bash

Sed

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, Texinfo**Test suite depends on:** Diffutils, Gawk**Must be installed before:** E2fsprogs, File, Libtool, Shadow

Shadow

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Sysklogd

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make, Patch**Test suite depends on:** No testsuite available**Must be installed before:** None

Sysvinit

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make, Sed**Test suite depends on:** No testsuite available**Must be installed before:** None

Tar

Installation depends on: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Patch, Sed, Texinfo**Test suite depends on:** Diffutils, Findutils, Gawk**Must be installed before:** None

Tcl

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Sed**Test suite depends on:** None**Must be installed before:** None

Texinfo

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, Sed**Test suite depends on:** None**Must be installed before:** None

Udev

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make**Test suite depends on:** Findutils, Perl, Sed**Must be installed before:** None

Util-Linux

Installation depends on: Bash, Binutils, Coreutils, E2fprogs, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, Sed, Zlib**Test suite depends on:** No testsuite available**Must be installed before:** None

Vim

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Sed**Test suite depends on:** None**Must be installed before:** None

Zlib

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed**Test suite depends on:** None**Must be installed before:** File, Module-Init-Tools, Util-Linux

长索引

软件包

程序

库

脚本

其他